

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
STROJNÍCKA FAKULTA**

**LINKSHIELD: MINIATURIZOVANÝ EXPERIMENTÁLNY  
MODUL NA SIMULÁCIU TLMENIA VIBRÁCIÍ NA  
ROBOTICKÝCH MANIPULÁTOROCH**

Bakalárska práca

SjF-13432-87701

2020

Martin Vríčan

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE  
STROJNÍCKA FAKULTA**

**LINKSHIELD: MINIATURIZOVANÝ EXPERIMENTÁLNY  
MODUL NA SIMULÁCIU TLMENIA VIBRÁCIÍ NA  
ROBOTICKÝCH MANIPULÁTOROCH**

**Bakalárska práca**

**SjF-13432-87701**

Študijný odbor:	Automatizácia strojov a procesov
Študijný program:	kybernetika
Školiace pracovisko:	Ústav automatizácie, merania a aplikovanej informatiky
Vedúci záverečnej práce:	doc. Ing. Gergely Takács, PhD.
Konzultant:	Ing. Erik Mikuláš

**Bratislava, 2020**

**Martin Vríčan**



## ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Martin Vríčan**  
ID študenta: 87701  
Študijný program: automatizácia a informatizácia strojov a procesov  
Študijný odbor: kybernetika  
Vedúci práce: doc. Ing. Gergely Takács, PhD.  
Konzultant: Ing. Erik Mikuláš  
Miesto vypracovania: ÚAMAI SjF STU v Bratislave

Názov práce: **LinkShield: Miniaturizovaný experimentálny modul na simuláciu tlmenia vibrácií na robotických manipulátoroch**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Úlohou študenta je navrhnuť a vyrobiť miniaturizovaný experimentálny modul na tlmenie vibrácií na tenkých robotických manipulátoroch. Prístroj bude kompatibilný s elektronickým rozložením Arduino R3 a bude sa volať „LinkShield“. Programátorské rozhranie bude písat v jazyku C/C++ pre prostredie Arduino IDE. Študent taktiež vytvorí rôzne inštruktážne príklady na spätnoväzobné riadenie tohto systému.

V rámci bakalárskej práce študent musí

- navrhnuť experimentálne zariadenie, vybrať vhodné elektronické a mechanické komponenty,
- navrhnuť elektrické zapojenie a plošnú obvodovú dosku,
- vyrobiť a otestovať funkčnosť prvej verzie modulu,
- napísať programátorské rozhranie (application programming interface, API) pre ovládanie zariadenia,
- vytvoriť inštruktážne príklady vhodné na didaktické nasadenie (napr. identifikácia, spätnoväzobné riadenie),
- využiť prostredie GitHub na manažovanie verzií softvéru.

Rozsah práce: 30-40 s.

Riešenie zadania práce od: 17. 02. 2020

Dátum odovzdania práce: 19. 06. 2020

**Martin Vríčan**  
študent

**prof. Ing. Cyril Belavý, CSc.**  
vedúci pracoviska

**prof. Ing. Cyril Belavý, CSc.**  
garant študijného programu

## Čestné prehlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej literatúry.

Bratislava, 16. júna 2020

.....  
Vlastnoručný podpis

Najväčšia vďaka patrí vedúcemu mojej práce doc. Ing. Gergelymu Takácsovi PhD. za neustálu motiváciu odborné rady a konštruktívnu kritiku, vďaka ktorým som mohol napredovať v mojej práci a naučiť sa mnoho nových vecí. Ďalej by som rád podľakoval konzultantovi práce Ing. Erikovi Mikulášovi za rady v oblasti elektroniky a 3D tlače, bez ktorej by sa hľadali riešenia technického dizajnu komplikovanejšie. V neposlednom rade by som rád podľakoval rodine a priateľom, ktorí ma psychicky podporovali a priateľke Patke, vďaka ktorej viem, že ak niečo pochopí tak je to napísané dobre.

Bratislava, 16. júna 2020

Martin Vríčan

**Názov práce:** LinkShield: Miniaturizovaný experimentálny modul na simuláciu tlmenia vibrácií na robotických manipulátoroch

**Kľúčové slová:** tlmenie vibrácií, robotické rameno, LinkShield, Arduino, spätnoväzbové riadenie, PPF riadenie

**Abstrakt:** V práci je opísaný experimentálny modul na simuláciu tlmenia vibrácií robotického ramena. Modul je tvorený pre didaktické účely. V kapitolách je postupne opísaný návrh modulu, výber komponentov a témy, ktoré s nimi súvisia. Modul je navrhovaný ako nadstavba pre prototypizačnú dosku Arduino. Ďalej je v práci opísaná tvorba programového rozhrania, ktoré obsahuje hlavné funkcie pre ovládanie modulu. Pre modul sú vytvorené dva inštruktážne príklady, ktoré ukazujú jeho funkčnosť. V jednom príklade je používané spätnoväzobné riadenie, ktoré znázorňuje jednu z možností tlmenia vibrácií. V závere práce sú zhrnuté výsledky testovania a možné úpravy hardvéru. Prílohy obsahujú zdrojové kódy pre knižnicu a inštruktážne príklady.

**Title:** LinkShield: Miniaturized Experimental Module for Simulating Vibration Damping on Robotic Manipulators

**Key words:** vibration damping, robotic arm, LinkShield, Arduino, feedback control, PPF control

**Abstract:** This work describes an experimental module to simulate vibration damping for emulated robotic arms. The module is created for didactic purposes. The chapters gradually describe hardware design of the module, the selection of components and the problems associated with them. The module is proposed as an extension for the family of Arduino prototyping board. Furthermore, this thesis describes the application program interface, which contains the main functions for controlling the module. Two instructional examples are created for the module, which showcase its functionality. In one example feedback control is used, which illustrates one of the vibration damping algorithms. At the end of the work the results are summarized describing the experiments and possible hardware modifications. The appendices contain the source code for the library and instructional examples.

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Hardvér</b>	<b>3</b>
1.1 Návrh elektronického obvodu a výber hardvéru . . . . .	3
1.1.1 Servomotor . . . . .	3
1.1.2 Prúdová a spätnoprúdová ochrana . . . . .	5
1.1.3 Akcelerometer . . . . .	6
1.1.4 Prepojovacie vodiče a konektory . . . . .	10
1.1.5 Rameno . . . . .	10
1.1.6 Potenciometer . . . . .	10
1.1.7 Návrh obvodu . . . . .	10
1.2 Obvodová doska . . . . .	11
1.2.1 Návrh PCB . . . . .	11
1.3 Upevnenie ramena . . . . .	13
1.4 Vyhotovenie . . . . .	14
1.4.1 Pájkovanie . . . . .	14
1.4.2 Servomotor, rameno a snímač . . . . .	15
1.5 Cena . . . . .	16
<b>2 Programové rozhranie</b>	<b>18</b>
2.1 Hlavičkový súbor . . . . .	19
2.1.1 Knižnice . . . . .	20
2.1.2 Definície . . . . .	20
2.1.3 Triedy . . . . .	21
2.1.4 Objekty . . . . .	22
2.2 Metódy . . . . .	22
2.2.1 Inicializácia . . . . .	22
2.2.2 Metódy snímača . . . . .	23
2.2.3 Kalibrácia . . . . .	26
2.2.4 Verejné metódy . . . . .	26
<b>3 Identifikácia a riadenie</b>	<b>28</b>
3.1 Zber dát pre identifikáciu . . . . .	28
3.1.1 Modelovanie . . . . .	30
3.1.2 Identifikácia parametrov prenosovej funkcie . . . . .	31
3.2 PPF . . . . .	31

<b>4 Záver</b>	<b>37</b>
<b>Literatúra</b>	<b>38</b>
<b>Dodatok A Zdrojový kód programového rozhrania</b>	<b>40</b>
<b>Dodatok B Identifikácia</b>	<b>43</b>
<b>Dodatok C PPF regulácia</b>	<b>45</b>
<b>Dodatok D Držiak</b>	<b>46</b>

## Zoznam skratiek

ADC	Analógovo číslicový prevodník (z angl. Analog Digital Converter)
API	Rozhranie pre programovanie aplikácií (z angl. Application Programming Interface)
AREF	Analógová referencia (z angl. Analog Reference)
BEMF	Spätná elektromotorická sila (z angl. Back Electromotive Force)
CS	Voľba čipu (z angl. Chip Select)
EMI	Elektromagnetická indukcia (z angl. Electromagnetic Induction)
FFC	Flexibilný plochý kábel (z angl. Flaxible Flat Cable)
IMU	Inerciálna meracia jednotka (z angl. Inertial Measurement Unit)
MEMS	Mikro elektro mechanický systém (z angl. Micro Electromechanical System)
PCB	Tlačená obvodová doska (z angl. Printed Circuit Board)
PID	proporcionálne integračne derivačný regulátor
PPTC	Polymérová poistka s pozitívnym teplotným koeficientom (z angl. Polymeric Positive Temperature Coefficient device)
SMD	Komponent na povrchovú montáž (z angl. Surface Mount Device)
PWM	Impulzová šírková modulácia (z angl. Pulse Width Modulation)
LSB	Najmenej významný bit (z angl. Least Significant Bit)

## Zoznam premenných a veličín

Symbol	Premenná alebo veličina	Jednotka v sústave SI
$b$	tlmiací koeficient	$\text{kg}\cdot\text{s}^{-1}$
$b_c$	kritický tlmiací koeficient	$\text{kg}\cdot\text{s}^{-1}$
$c$	konštantá lineárne pôsobiaceho akčného člena	$\text{rad}^{-2}$
$C$	konštantá lineárne pôsobiaceho akčného člena	$\text{kg}\cdot\text{s}^{-2}$
$F(t)$	sila pôsobiaca na systém	N
$g$	parameter zosilnenia	-
$G(s)$	obrazový prenos	-
$k$	pružinová konštantá	$\text{N}\cdot\text{m}^{-1}$
$m$	hmotnosť	$\text{kg} \text{ (g)}$
$\omega$	uhlová frekvencia	$\text{rad}\cdot\text{s}^{-1}$
$q(t)$	výstup zo systému	závisí od veličiny
$r$	referenčná hodnota	°
$s$	argument laplaceovej transformácie	-
$t$	čas	s
$T_s$	vzorkovacia periódna	s (ms)
$u$	výstup z regulátora	závisí od veličiny
$u_r$	regulovaný vstup do systému	závisí od veličiny
$u(t)$	vstup do systému	závisí od veličiny
$y(t)$	časovo zavislý člen	závisí od veličiny
$z$	vzorka	-
$\zeta$	tlmiací pomer	-

# Úvod

Používanie priemyselných robotických ramien a manipulátorov je nevyhnutná súčasť automatizácie výrobných procesov. Ich cieľom je zrýchliť, zjednodušiť a zefektívniť výrobný proces. Ramená a manipulátory pozostávajú z tuhých masívnych štruktúr s výkonnými motormi. Pri manipulácii je potrebná vysoká presnosť a dosiahnutie čo najväčšej rýchlosťi. Tuhá štruktúra je podstatná vlastnosť pri presnosti polohy, avšak s požiadavkami na maximálnu nosnosť narastá hmotnosť ramien. V článku zameranom na riadenie vibrácií robotických manipulátorov z roku 2011 autori [9] uvádzajú, že dodnes väčšina robotov dokáže manipulovať s objektami nie ľažšími ako 5% ich vlastnej hmotnosti, aby sa minimalizovali štrukturálne vibrácie. Veľkosť vibrácií závisí aj od veľkosti zrýchlenia najmä pri zastavovaní. V praxi je to kompenzované nižšími rýchlosťami pohybov alebo plynulejším dobrzdovaním, čo vplýva na dĺžku cyklu premiestňovania.

Mnoho zdrojov z literatúry sa zaobera urýchlením cyklu, ktoré sú založené na skúmaní dynamických vlastností manipulátorov. V článku [10] publikovanom na medzinárodnej konferencii o strojovom učení a kybernetike je opísaný experiment pre riadenie jednoduchého ramena, kde bola skombinovaná metóda lineárizácie výstupu so spätnou väzbou a vstupné tvarovanie, čo malo za následok úspešné zníženie vibrácií.

Autori [9] sa pokúšajú redukovať vibrácie na dvojramennom manipulátore prostredníctvom PID regulátora. Pri experimente bol cieľ ustabilizovať podávač vo vodorovnej polohe kde manipulátor využíva len 2 zo 6 osí. Výstupné údaje sú získavané z akcelerometra vo forme zrýchlenia a z motora vo forme uhlu natočenia. Tento spôsob riadenia je vhodný len pre konkrétné prípady, kedy sa manipulátor pohybuje len v dvoch osiach a nenatáča sa.

Z experimentov je zrejmé, že táto problematika vyžaduje viac pozornosti. Pre účely experimentov sú vyrábané rôzne didaktické modely ktoré reprezentujú takéto systémy. Napríklad model od firmy Quanser [12] s názvom Rotary Flexible Link zobrazený na Obr. 1 umožnuje testovať rôzne metódy regulácie systému na základe rôznych výstupných dát. Takéto modely sú súčasťou užitočné pre didaktické účely, ale pre mnohé inštitúcie sú ľažko dostupné z hľadiska finančnej náročnosti. Ich cena môže dosahovať až niekoľko 10-tisíc eur.

Mojim cieľom je vytvoriť didaktickú pomôcku s názvom LinkShield, ktorá má slúžiť ako alternatíva ku kommerčným produktom za výrazne nižšiu cenu ako napríklad model jednoramenného robotického pružného ramena z Istanbulskej Technickej univerzity [6], ktorého výrobné náklady sú približne 90€. Pre tento model však nie je dostupný žiadna technická dokumentácia, v ktorom by bolo opísané ako je možné ho vyrobiť. V podstate bol tvorený len pre jeden konkrétny experiment.

LinkShield má napodobňovať tiež len jeden typ systému, ale bude mať kompletnú tech-



Obr. 1: Komerčný modul pre simulovaie vybrácií ramena, prevzaté z [12].

nickú dokumentáciu, ktorá postačí pre zostavenie takéhoto modulu komukolvek. Modul má napodobňovať správanie sa pružného robotického ramena, ktoré bude reprezentované tenkým úzkym pružným plechom. Ten bude upevnený pomocou držiaka na modelárskom servomotore, ktorý bude zakomponovaný priamo do obvodovej dosky. Výstupné údaje zo systému budú získavané napríklad z akcelerometra umiestnenom na konci ramena. Do systému budú vstupovať údaje o natočení servomotoru. Otáčaním servomotoru budú vznikať vibrácie ramena, ktoré bude cieľom utlmit.

Podobných miniatúrnych experimentov ako je LinkShield existuje už viacerô. Sú súčasťou nekomerčného didaktického projektu AutomationShield [5], medzi ktoré sa v prípade funkčnosti bude radíť aj LinkShield. Každý z modulov je navrhnutý pre napodobnenie systému v ktorom je možne aplikovať rôzne druhy riadení. V prípade LinkShieldu sa jedná o malý experiment, ktorého účelom je napodobiť správanie sa pružného manipulačného ramena. Modul je navrhovaný pre kompatibilitu s viacerými verziami mikropočítačov Arduino, ktoré riadia celý systém ako napríklad: UNO, LEONARDO a ZERO [3]. Pre každý modul je navrhnutá vlastná obvodová doska so zoznamom komponentov, ktoré sú voľne dostupné na platforme GitHub. Používateľ okrem programovania má aj možnosť vlastnoručne zostaviť hardvér, čo môže byť brané tiež ako forma praktickej výučby.

Návrh celého modulu pozostáva z návrhu obvodu, do ktorého je potrebné zvoliť optimálne komponenty tak, aby cena bola relatívne malá, ale modul dostatočne funkčný. Druhou časťou je tvorenie programového rozhrania API (z angl. Application Programming Interface - rozhranie pre programovanie aplikácií). To obsahuje potrebné funkcie, ktoré postačujú na ovládanie hardvéru. Projekt AutomationShield má určité zásady pre tvorbu programu, ktorých je dobré sa držať. Jedná sa najmä o podobnosť programového rozhrania, aby bolo jednoduché pracovať s viacerými modulmi naraz. Práca je zameraná na vytvorenie prvej funkčnej verzie LinkShieldu.

# 1 Hardvér

## 1.1 Návrh elektronického obvodu a výber hardvéru

V tejto podkapitole podrobne vysvetlím obvodovú schému môjho návrhu LinkShieldu a vysvetlím hlavné princípy fungovania jednotlivých komponentov a samotnej zostavy. Táto zostava sa skladá z dvoch častí z toho dôvodu, že vo fyzickej forme ide o dve obvodové dosky, ktoré medzi sebou komunikujú.

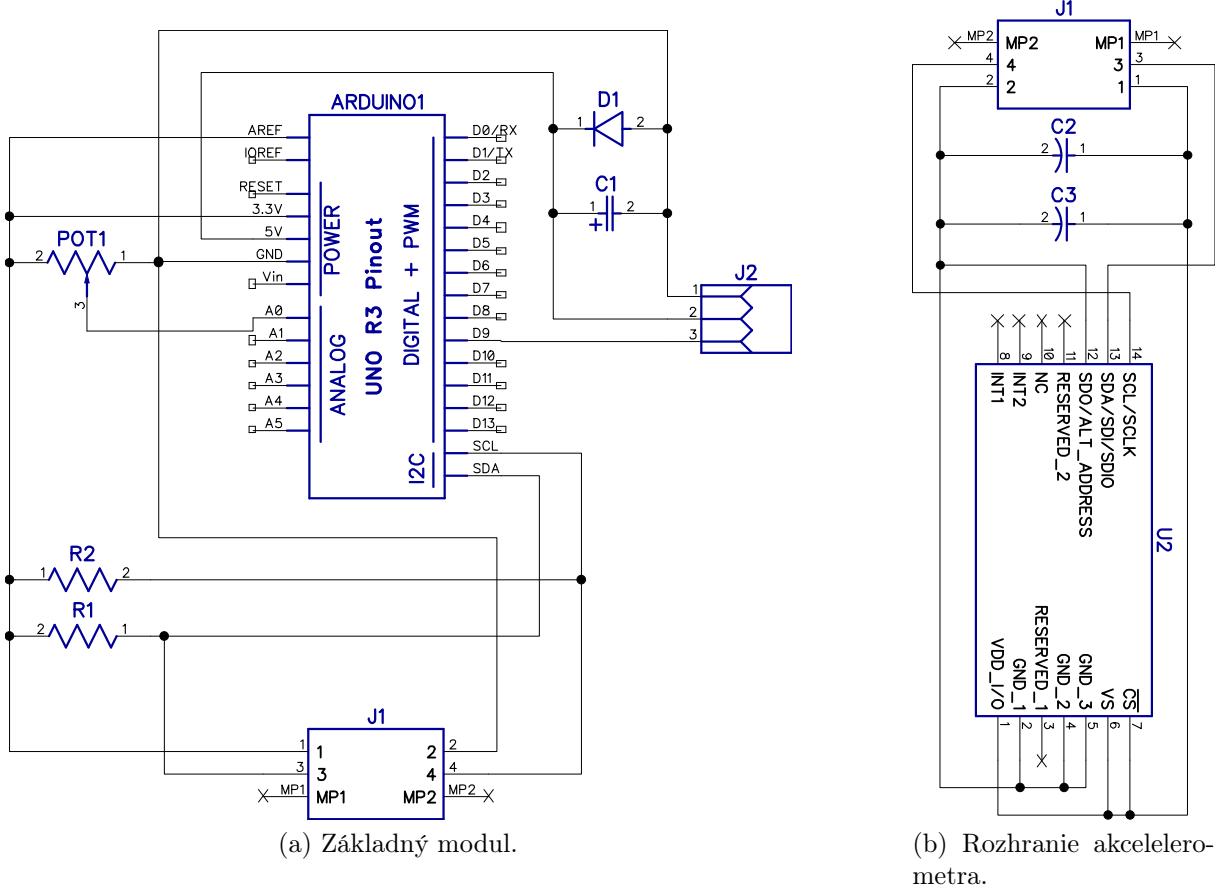
Na Obr. 1.1a je zobrazená schéma zapojenia hlavnej dosky, respektíve základne, ktorá je umiestnená priamo na konektoroch (z angl. pin) dosky Arduino. V tomto obvode je zobrazený mikropočítač ATmega328P [3]. Jedná sa o model UNO Rev3, ktorý je tu nevyhnutný preto, že jeho periféria sú podstatné pre návrh zvyšného obvodu a zároveň reprezentujú aj piny, s ktorými sú spojené jednotlivé komponenty. Ďalšími dôležitými súčasťami schémy sú tzv. pullup rezistory, potenciometer, konektory pre servomotor a snímač, dióda a kondenzátor.

Schéma pre druhú malú dosku je zobrazená na Obr. 1.1b. Hlavným komponentom tejto schémy je snímač zrýchlenia Analog Devices ADXL345 [2]. Spolu s ním sa tu nachádzajú dva kondenzátory a konektor pre komunikáciu s hlavnou doskou.

V obvode je zobrazené aj prepojenie napájania na 3.3 V s rozhraním AREF (z angl. Analog Reference) čo označuje referenčné napätie na analógových vstupoch. Toto prepojenie slúži na konfiguráciu analógových pinov na dané napätie v prípade, že vnútorný obvod niektorého snímača alebo zariadení podporuje 3.3 V logiku. V praxi to znamená, že očakávané hodnoty na vstupoch sú v rozsahu 0 - 3.3 V. Týmto nastavením, v prípade 5 V predvoleného rozhrania, zvýšime aj rozlíšenie analógovo-číslicového prevodníka ADC (z angl. Analog Digital Converter) na úkor rozsahu.

### 1.1.1 Servomotor

Dôležitou súčasťou celého modulu je modelársky servomotor, ktorý bude pri testovaní a programovaní vystupovať ako akčný člen. Výber správneho servomotora bol komplikovaný z hľadiska požiadaviek na parametre, ale súčasne aj na nízku cenu. Základné technické parametre ktoré sa udávajú pri servomotoroch sú: rýchlosť, moment sily, napájacie napätie, typ prevodov a veľkosť. Taktiež sú rozdelené do dvoch kategórií na polohové, na ktorých sa pri programovaní určuje uhol otočenia a spojité alebo rýchlostné, na ktorých sa ovláda rýchlosť. Pre účely LinkShieldu som vybral polohový servomotor z toho dôvodu, že pri programovaní je potrebné dosiahnuť čo najrýchlejšiu skokovú zmenu dosiahnutím najvyššej rýchlosťi a následnom zastavení, pri čom nie je potrebné regulovať rýchlosť. Taktiež nie je potrebný veľký rozsah otáčania. Pre účely modulu postačuje rozmedzie  $20^\circ$



Obr. 1.1: Elektronické schémy zapojení.

až  $\tilde{180}^\circ$ . Obmedzenie otáčania zároveň poskytne ochranu pre vodič, ktorý prepája Arduino a snímač, pretože ten by sa pri viacnásobnom otočení mohol poškodiť. Pri vyberaní ideálnej veľkosti, vzhľadom na rozmery modulu, bola jednoznačná veľkosť mikro. Ponuka servomotorov v tejto veľkosti je široká a ich rozmery sú ideálne pre zakomponovanie do modulu. Z hľadiska momentu výber neboli až taký dôležitý, pretože rameno ktorým bude servomotor otáčať nemá veľkú hmotnosť a rozdiely medzi rôznymi servomotormi z nižzej cenovej kategórie sú minimálne. Táto vlastnosť sa v technických špecifikáciách uvádzajú v  $(kg \times cm)$ . Podstatný parameter je rýchlosť otáčania, ktorá je dôležitá pre veľkosť výchylky po zastavení a vytvára tým pádom aj väčší vizuálny efekt. Rýchlosť servomotora sa udáva ako čas, za ktorý sa otočí o určitý uhol, najčastejšie o  $60^\circ$  ( $s^{-1}$ ). Dôležité je prihliadnuť aj na napájacie napätie (V). Pre napájanie priamo z dosky Arduino bez externého zdroja sú k dispozícii dve hodnoty napäcia a to 3,3 V a 5 V. Štandardné malé servomotory majú napájanie v rozmedzí od 4,8 V do 6 V. Prúdový odber a výkon servomotoru nie je vždy udávaný v technických špecifikáciách a zistiť ho je možné len meraním alebo testovaním priamo na doske. Aby sa dosiahla čo najpresnejšia poloha bez vôle, rozhodol som sa vybrať pohybový mechanizmus s kovovými prevodmi. Ich výhoda je aj v nižšom opotrebovaní a vyššej tuhosti. Takisto bolo dôležité, aby hlavný čap bol správne uchytený na ložisku, aby nevznikala výchylka na osi otáčania vplyvom rýchleho opotrebenia. Rýchlosť odozvy je tiež dôležité nezanedbať. Keďže rameno má malú dĺžku, bude vibrovať s pomerne vysokou

frekvenciou, na ktorú nemusí byť servomotor schopný reagovať.

### Analógový servomotor K-POWER - MB0090

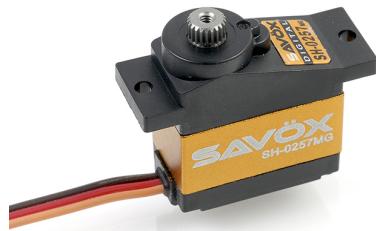
MB0090 je model servomotoru zobrazený na Obr. 1.2a, ktorý som vybral pre prvé testovanie LinkShieldu. Rozsah napájacieho napätia má od 4,8 V do 6 V. V tomto prípade je napájané z 5 V pinu priamo z Arduina. Vnútorné prevody sú vyrobené z kovu a horný čap je usadený v guľôčkovom ložisku. Servomotor má v sebe zakomponovanú analógovú spätnú väzbu. Pri testoch sa však zistilo, že tento servomotor nedisponuje dostatočne rýchlu odozvou, ktorá by bola potrebná pre dostatočne jemné riadenie. Zároveň nepresnosť analógovej spätnej väzby spôsobovala rozkmitanie pri ustálených stavoch. Po porovnaní vlastností som sa rozhodol vybrať model, ktorý obsahuje digitálnu spätnú väzbu. Okrem rýchlejšej odozvy dokáže presnejšie udržiavať zadanú polohu. Tieto vlastnosti však idú na úkor ceny [4].

### Číslicový servomotor Savox - SH-0257MG

Ako druhý servomotor bol testovaný model od firmy Savox SH-0257MG, ktorý je na Obr. 1.2b. Jedná sa o digitálny mikro servomotor s napájacím napäťom od 4,8 V do 6 V podobne ako vyššie spomínaný model. Taktiež sa vo vnútri nachádzajú kovové prevody a jedno guľôčkové ložisko. Keďže predošlý servomotor má zhodné rozmery aj napájanie, môže byť bezproblémovo nahradený. Pri 5 V dokáže vyvinúť rýchlosť 0,13 s/60° a moment 1,9 kg/cm, čo sú v porovnaní sa analógovým servomotorom horšie vlastnosti, ale vzhľadom na cenu digitálnych s rovnakými vlastnosťami, ktoré sú mnohonásobne vyššie, bolo nutné vybrať slabší model, avšak kľúčovú vlastnosť, ktorú bolo treba dosiahnuť, rýchlejšiu odozvu, splňa. Pri testovaní modelu dokázal dostatočne rýchlo rozkmitať rameno a včas reagovať na zmeny polohy.



(a) MB0090 [7]



(b) SH-0257MG [13]

Obr. 1.2: Testované servomotory.

#### 1.1.2 Prúdová a spätnoprúdová ochrana

Pri prvotných testoch sa mnohokrát stávalo, že po spustení servomotora nastalo vypnutie mikrokontroléra Arduino. Tento jav nastal z dôvodu prechodového stavu motora, kde dochádza k prudkému nárastu elektrického prúdu, pri ktorom zaúčinkuje prúdová ochrana

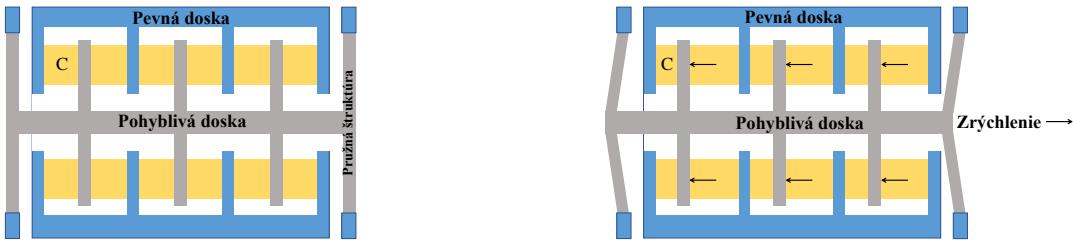
na doske mikrokontroléra vo forme vratnej poistky. Táto poistka je pasívna elektronická súčiastka umiestnená na doske Arduina v blízkosti napájania aby sa predišlo poškodeniu akýchkoľvek komponentov. Je označovaná skratkou PPTC (z anglického Polymeric Positive Temperature Coefficient device - polymérová poistka s pozitívnym teplotným koeficientom) [14]. Ako už zo skratky vyplýva, obsahuje vodivý polymérový materiál, ktorý ochrani obvod zmenou odporu pri prúdovom prechádzaní z nízkeho na vysoký. Počas stavu s vysokým odporom cez poistku neustále prechádza malý prúd, ktorý udržiava tento stav ale nie je schopný napájať mikrokontrolér, čím zabráňuje jeho zničeniu. Poistka sa vráti do pôvodného stavu keď sa chyba odstráni, čím klesne prúd, ktorý ňou prechádza alebo zariadenie sa odpojí od zdroja napájania. Keďže pri zaúčinkovaní tejto poistky sa nedokážeme vyhnúť odpojeniu a resetovaniu mikrokontroléra, je potrebné použiť iný prvok, ktorý vykompenzuje spomínané prúdové špičky. Je potrebné podotknúť, že dosku chceme napájať len USB portom bez nutnosti použitia externého zdroju v záujme zníženia ceny. Ako riešenie sa zvykne využívať paralelne zapojený kondenzátor. Vo všeobecnosti okrem iného slúžia predovšetkým ako zásobníky elektrickej energie, ktorou kompenzujú nárasty a poklesy prúdov v obvodoch. V tomto prípade som zvolil tantalový kondenzátor, ktorý sa vyznačuje nižším stratovým odporom a indukčnosťou. Výrába sa s malými rozmermi ako SMD (z angl. Surface Mount Device - komponent na povrchovú montáž). Kapacita zvoleného kondenzátora je  $22 \mu\text{F}$  a maximálne napätie  $12 \text{ V}$ .

Do obvodu spolu s kondenzátorom je paralelne pridaná aj dióda, ktorá má za úlohu ochrániť obvod pred spätnou elektromotorickou silou BEMF (z angl. Back Electromotive Force). Tento jav nastáva pri zmene prúdu na cievke alebo induktore, pri čom dochádza k zmene magnetického poľa a tým pádom samoindukovaniu napäcia.

### 1.1.3 Akcelerometer

Akcelerometre sú komponenty, ktoré slúžia na meranie gravitačného aj dynamického zrýchlenia. Ich využitie v elektronike je rozsiahle, napríklad na detekciu orientácie zariadenia voči zemskému povrchu v mobilných zariadeniach za účelom preklápania obrazu, taktiež vo fotoaparátoch a kamerách, kde slúži na stabilizáciu obrazu, v pevných diskoch na zaistenie ochrany pred pádom, v herných ovládačoch, navigačných systémoch a mnogo iných. Ich princíp je založený na premene mechanickej energie na elektrickú, ktorá je ďalej spracovávaná prostredníctvom mikrokontroléra. Sú pripojené priamo na objekte, resp. mieste, na ktorom je potrebné merať zrýchlenie. Rozsah sa spravidla udáva v jednotkách gravitačnej sily ( $\text{g}$ ). V prípade, že na akcelerometer nepôsobí vonkajšie zrýchlenie, vektorovým súčtom dostaneme vždy hodnotu zrýchlenia  $g = 1$  kolmo k zemskému povrchu. Akcelerometre sa vyrábajú pre rôzne veľkosti zrýchlení, v rozsahoch od  $1,5 \text{ g}$ , až do  $400 \text{ g}$ . Existuje viacero typov senzorov, medzi najpoužívanejšie patria akcelerometre so štruktúrou MEMS (z anglického Micro-Electro-Mechanical Systems), ktoré som sa rozhodol používať na LinkShielde. Ďalšie typy akcelerometrov sú piezoelektrické a tepelné.

Akcelerometre so štruktúrou MEMS fungujú na princípe zmeny kapacity alebo odporu vnútorného obvodu [8]. Na Obr. 1.3 je znázorený základný princíp, kde môžeme vidieť pohyblivú dosku alebo závažie, ktoré je upevnené na pružných prvkoch. To sa pri pôsobení sily vychýľuje a mení vzdialenosť medzi pevnou doskou, čím sa mení kapacita  $C$ . Tento typ akcelerometrov je ľahko dostupný s malými rozmermi, z toho dôvodu som sa rozhodol zúžiť výber na túto kategóriu.



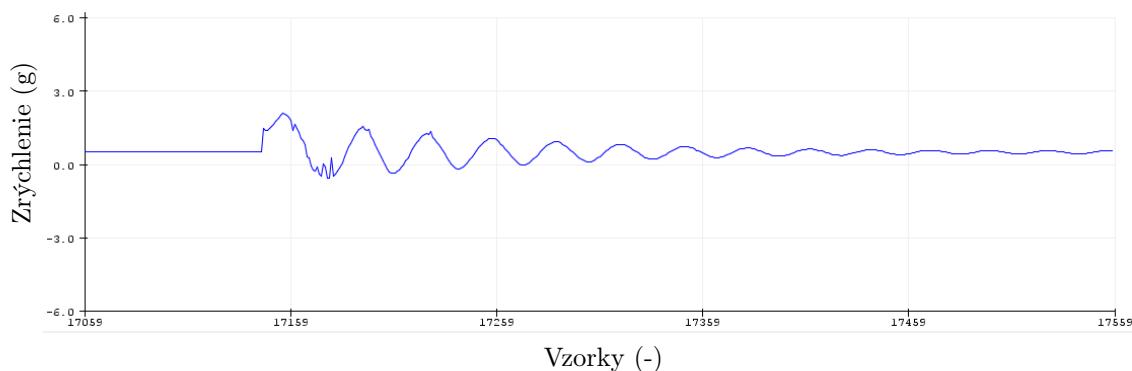
Obr. 1.3: Zjednodušená schéma akcelerometra MEMS

Na komunikáciu s mikrokontrolérom využívajú rôzne metódy. Najjednoduchšia metóda komunikácie je pomocou analógového výstupu, pri ktorom sú hodnoty zrýchlení priamoúmerné výstupnému napätiu. Pre každú os využíva senzor jeden analógový výstup. Rozsah týchto akcelerometrov je pevne daný od výrobcu. Rozlíšenie je závislé od rozlíšenia analógovo-číslicového prevodníka (ADC), ktorý premieňa analógový signál na číslicový. Takýto signál je mikrokontrolér už schopný spracovať. Arduino UNO má vo svojom mikrokontroléri zabudovaný AD prevodník, ktorý má 10-bitové rozlíšenie, čo znamená, že dokáže rozoznať  $2^{10}$  úrovní napäti. Ďalší typ komunikácie ktorý sa často využíva je prostredníctvom rozhrania pre sériovú komunikáciu SPI alebo I2C. Výhodou pri takejto komunikácii oproti analógovej je, že rozlíšenie nezávisí od ADC prevodníka, ale od rozlíšenia samotného senzora. Taktiež sa tieto akcelerometre môžu rôzne konfigurovať, ak to daný model podporuje. Pomocou nastavenia zabudovaných registrov sa môže nastavovať rozsah, odchýlka a mnoho ďalších parametrov.

### Analógový akcelerometer ADXL326

Pre prvotné testovanie za účelom zistenia najvyšších dosahovaných hodnôt zrýchlenia som zvolil analógový akcelerometer ADXL 326, ktorý je umiestnený na malej prototypizačnej obvodovej doske zobrazenej na Obr. 1.5a. Senzor je napájaný napäťom 3,3 V. V obvode je umiestnený napäťový regulátor z 5 V na 3,3 V, čo umožňuje napájať ho napäťom s obidvomi hodnotami. Obvod tiež obsahuje filtračné kondenzátory medzi napájacou vetvou a zemou a zhodne aj pri každom výstupe. Rozsah akcelerometra je  $\pm 16$  g. Špecifikovaný rozsah je garantovaný minimálny rozsah, čo znamená, že je schopný odmerať aj vyššie hodnoty ale nie je zaručená ich presnosť. Výstupný signál je v hodnotách od 0 V do 3,3 V, kde 0 V označuje -16 g a 3,3 V +16 g. Arduino má predvolené nastavenie referenčného napäťia na analógových vstupoch 5 V čo znamená, že pri používaní senzorov s nižšími referenčnými napäťami, ako napríklad spomínaný akcelerometer, sa nevyužíva plné rozlíšenie AD prevodníka. Na dosiahnutie plného rozlíšenia je potrebné nastaviť referenčné napätie na 3,3 V. Pre účely testovania postačuje zaznamenávať hodnoty z jednej osi, ktorá je rovnobežná so smerom pohybu ramena. Dôležité je umiestniť senzor na čo najvzdialenejší bod ramena od osi otáčania, kde dochádza k najväčšej výchylke. Na Obr. 1.4 je zobrazený priebeh nasnímaných hodnôt zrýchlení pri otočení servomotorom o  $180^\circ$ . Tieto hodnoty sú len orientačné, pretože sa jedná o prvotný test s provizórny hardvérom, na ktorom boli použité štandardné prepojovacie vodiče, ktoré v značnej miere pôsobili ako tlmič vibrácií. Na obrázku je možné vidieť ostré zmeny hodnôt, ktoré neodpovedajú skutočnému

riebehu. Po opakovaných testoch som zistil, že môže ísiť o napäťové poruchy, ktoré vznikajú pri prechodnom stave servomotoru. Toto tvrdenie podporuje aj test na dvoch rôznych mikrokontroléroch, kde servomotor bol ovládaný prvým a hodnoty boli čítané na druhom. Pri týchto meraniach sa spomínané výchylky už nenachádzali. Pri pokuse o vyhľadenie týchto zmien som používal hliníkové elektrolytické kondenzátory zapojené paralelne na motor s hodnotami od  $10 \mu\text{F}$  do  $50 \mu\text{F}$ , ktoré neboli vo veľkej miere účinné. Vzhľadom na tieto výsledky som sa rozhodol otestovať aj akcelerometer s digitálnym výstupom.



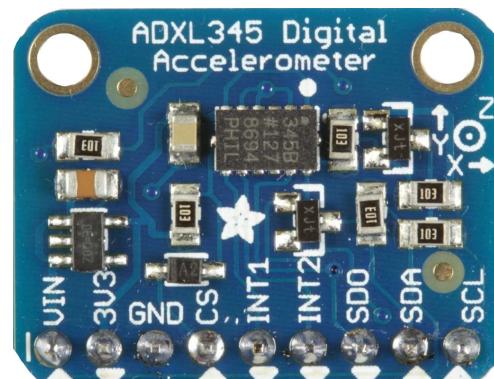
Obr. 1.4: Meranie pomocou analógového akcelerometra.

### Číslicový akcelerometer ADXL345

ADXL je malý 3-osový číslicový akcelerometer s vysokým rozlíšením 13-bitov do  $\pm 16$  g. Najvyššia citlivosť snímača je  $0,004 \text{ g/LSB}$  (z angl. Least Significant Bit - Najmenej Významný Bit). Ako u predchádzajúceho modelu aj tento je dostupný ako malý prototypizačný obvod zobrazený na Obr. 1.5b, v ktorom je zabudovaný napäťový regulátor a kondenzátory. Pre jeho veľké rozmery nie je vhodný pre používanie na LinkShield a je potrebné si vytvoriť vlastný menší obvod, ale pre testovanie na improvizovanom hardvéri postačuje. Rozmery samotného snímača sú  $5 \times 3 \times 1 \text{ mm}$ .



(a) ADXL326 [1]



(b) ADXL345 [2]

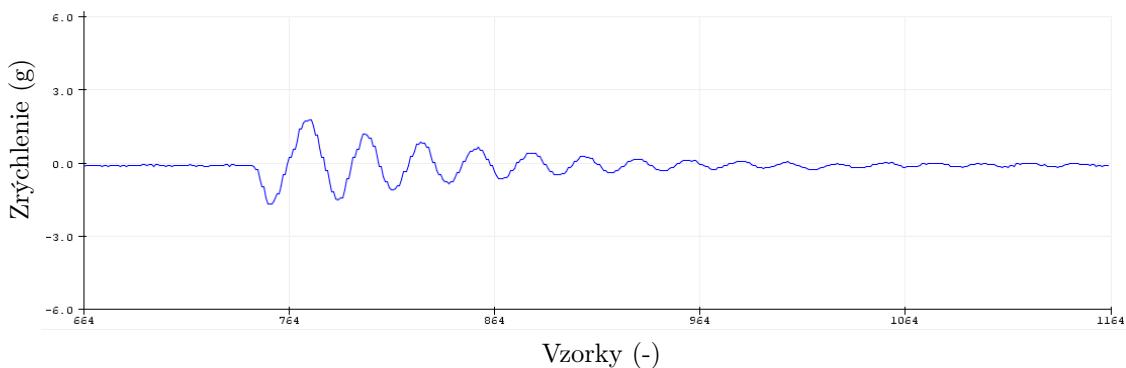
Obr. 1.5: Používané akcelerometry.

Tento číslicový akcelerometer obsahuje niekoľko užitočných funkcií. Umožňuje užívateľskú voľbu rozsahu na  $\pm 2$  g,  $\pm 4$  g,  $\pm 8$  g, alebo  $\pm 16$  g. Taktiež je možné nastaviť rýchlosť vzorkovania v rozmedzí od 0,1 Hz až do 3200 Hz. V prípade potreby je možné akcelerometer priamo kalibrovať nastavením odchýlky pre každú os zvlášť, ktorá sa pripočítá ku každej výstupnej hodnote. Táto funkcia môže ušetriť časť výpočtovej pamäte a výkonu mikrokontroléra. Na komunikáciu s mikrokontrolérom je možné využívať rozhrania I2C alebo SPI. Výstupné dátá sú formátované pre každú os samostatne ako 16-bitový dvojkový doplnok, čo znamená, že hodnotu dostaneme sčítaním dvoch 8-bitových hodnôt.

V prípade LinkShieldu som zvolil komunikáciu prostredníctvom zbernice I2C. Pre správne čítanie hodnôt s touto komunikáciou je potrebné do obvodu zapojiť pull-up rezistory, ktoré definujú logické hodnoty pri neurčitom stave. Napájajú sa medzi napájanie, v tomto prípade na 3,3 V a komunikačné žily SCL pre hodinový signál (R2) a SDA pre sériové dátá (R1). Hodnoty rezistorov sú  $10\text{ k}\Omega$ . Aby snímač vedel, že má očakávať komunikáciu prostredníctvom I2C, je na ňom vyvedený pin s označením CS (z angl. Chip Select - zvolenie čipu), ktorý musí byť pre túto komunikáciu napájaný napäťom digitálneho rozhrania, čo v našom prípade je 3,3 V. Zariadenia používajúce I2C komunikáciu majú predvolenú a alternatívnu adresu, ktorá sa vyberaná podľa napájania niektorých z pinov na snímačoch. V prípade zvoleného akcelerometra je pin pre voľbu adresy označený ako ALT\_ADDRESS. Na LinkShielde nie sú iné zariadenia, ktoré používajú tento typ komunikácie, preto nezáleží na výbere adresy. Avšak, aby bol stav pinu jednoznačný, je pripojený na zem. V prípade voľného ponechania by sa na pine mohlo naindukovať napätie, ktoré by spôsobilo zmenu adresy a tým pádom aj chybu pri komunikácii.

Pre správnu funkčnosť snímača výrobca odporúča do obvodu zapojiť paralelne tantalový kondenzátor s kapacitou  $1\text{ }\mu\text{F}$  (C2), ktorý je polarizovaný a musí byť zapojený katódou na zdroj a anódou na zem a keramický kondenzátor s kapacitou  $0,1\text{ }\mu\text{F}$  (C3), ktorý nie je polarizovaný. Slúžiť majú predovšetkým na oddelenie šumu na napájacom zdroji.

Pri testovaní s týmto snímačom sa už neprejavili náhle zmeny hodnôt, ako pri predchádzajúcim modely. Výsledky meraní je možné vidieť na Obr. 1.6 . Aby sa výsledky dali porovnať, otočenie servomotora bolo o rovnaký uhol ako pri predchádzajúcich testoch.



Obr. 1.6: Meranie pomocou digitálneho akcelerometra

#### **1.1.4 Prepojovacie vodiče a konektory**

Ako bolo vyššie spomínané, pri testoch sa vyskytoval problém s vodičmi, ktoré pôsobili ako tlmič vibrácií. Aby rameno nebolo tlmené pasívne, je potrebné použiť čo najtenšie resp. flexibilné vodiče, ktoré by tento efekt nespôsobovali. Najlepší variant, ktorý je ľahko dostupný a často využívaný, sú FFC vodiče (z angl. Flexible Flat Cable - flexibilný plochý vodič). Tieto vodiče sú vo forme tenkého pásiku, cez ktorý je vedených viac žil. V tomto prípade sú potrebné 4 žily. Na zapojenie FFC káblu sa používa špecifický konektor, ktorý rozmermi pinov odpovedá rozmerom káblu.

#### **1.1.5 Rameno**

Nevyhnutnou súčasťou LinkShieldu je rameno. Na účely emulácie dynamického charakteru robotickej ruky je potrebné, aby bolo dostatočne pružné, aby ho bolo možné rozkmitať. Zároveň nemôže byť príliš veľké, aby neprevažovalo celý LinkShield. Zvolené rameno je vyrobené z nerezového plechu s rozmermi  $10 \times 85 \times 0,2$  mm. Na jednom konci je vyvŕtaný otvor, ktorý slúži na upevnenie k držiaku.

#### **1.1.6 Potenciometer**

Posledným komponentom, ktorý sa na LinkShielde nachádza je potenciometer. Možnosť využitia je najmä pre ručné otáčanie servomotoru a zadávanie referenčnej polohy. Potenciometer je 3-pinový s maximálnym odporom  $10\text{ k}\Omega$ . Napájaný je z 3,3 V pinu. Druhý kontakt, na ktorom je konštantný odpor voči napájaniu, je uzemnený a posledný kontakt je vyvedený na analógový vstup. Aj v tomto prípade sa využije nastavenie referenčného napäťia na 3,3 V na zaručenie kompatibility s inými čipmi.

#### **1.1.7 Návrh obvodu**

Na návrh obvodu som používal softvér DipTrace. Je určený predovšetkým na tvorenie schematických diagramov a následne podľa nich na tvorenie tlačených obvodových dosiek PCB (z angl. Printed Circuit Board). Softvér sa pozostáva zo štyroch nástrojov, Schematic Capture na návrh obvodu, PCB Layout na návrh obvodovej dosky, Component Editor na tvorbu schematických značiek komponentov a Pattern Editor na tvorbu rozloženia pinov na doske. Pre účely LinkShieldu postačuje bezplatná verzia, ktorá umožňuje tvoriť schémy, ktoré obsahujú komponenty do 100 pinov a umožňujú tvoriť dvojvrstvovú obvodovú dosku.

Na návrh obvodu sa teda použije prvý nástroj, Schematic Capture. V inštalačnom balíku obsahuje knižnicu s množstvom komponentov, ku ktorým som pridal knižnicu AutomationShield, ktorá obsahuje ďalšie, často používané pri iných moduloch. Schematické zobrazenie a fyzické rozhranie komponentov je v programe navzájom prepojené, preto je potrebné už pri návrhu schémy vyberať komponenty s požadovanou veľkosťou. Pre FCC konektor a akcelerometer bolo potrebné vytvoriť vlastné schematické označenie a aj vzor pre fyzické rozloženie pinov a montážnych plôch.

Ako základ hlavnej schémy pre základňu modulu je použitý vzor mikrokontroléra Arduino UNO, ktoré je súčasťou knižnice AutomationShield a vo fyzickom zobrazení odpovedá rozmerom dosky Arduino s konektormi a montážnymi dierami. Následne postačovalo

priadať všetky komponenty a spojiť piny podľa stanovených požiadaviek a funkcií. Servo-motor je v schéme zobrazený len ako konektor, na ktorý sa pripoja vodiče.

Pri návrhu rozhrania pre akcelerometer poslúžil ako základ snímač, ku ktorému sa následne pridali nevyhnutné komponenty. Okrem FCC konektora do obvodu so senzorom je potrebné umiestniť kondenzátory spomínané v Kap. 1.1.3.

## 1.2 Obvodová doska

Obvodové dosky alebo dosky plošných spojov PCB (z angl. Printed Circuit Board) sú dosky používané v elektronike na vytváranie elektronických obvodov a následné upevnenie komponentov. Sú väčšinou vyrobené zo sklolaminátu, ktorý je nevodivý, na ktorom sú tenké medené cesty slúžiace ako vodiče. Na povrchu je nanesená ochranná alebo spájkovacia vrstva, ktorá chráni cesty pred vonkajšími vplyvmi, ktoré by ich mohli poškodiť alebo spôsobiť skrat a pomáha aj pri spájkovaní. Odhalené sú len miesta, kde sa v poslednej fáze pripievajú elektronické komponenty, ktoré sa na dosku upevňujú pomocou spájky, ktorá spoje elektronicky prepojí a udrží na mieste.

Komponenty môžeme rozdeliť do dvoch skupín na klasické s kovovými vývodmi vo forme drôtu alebo SMD komponenty (z angl. Surface Mount Device). Vývody na klasických komponentoch sa prestrčia cez predvŕtaný otvor a z opačnej strany dosky sa prispájkujú. Môžu byť umiestňované z oboch strán. SMD komponenty nemajú nožičky ani vývody, majú len kovové plôšky zo spodnej alebo bočnej strany, ktoré presne odpovedajú plôškam na obvodovej doske. Výhodou SMD proti klasickým komponentom je, že sa nemusí dávať pozor na možný kontakt s iným predmetom z opačnej strany, avšak ručné spájkovanie SMD je komplikovanejšie.

Obvodové dosky môžu byť jedno, dvoj alebo viacvrstvové. Počet vrstiev sa určuje z hľadiska náročnosti obvodu. Jednotlivé cesty na rôznych vrstvách sú prepojené pokovovanými prechodmi, tiež označované ako slepé prechody, je však lepšie keď ich je čo najmenej.

### 1.2.1 Návrh PCB

Na návrh som používal nástroj PCB Layout z aplikácie DipTrace. Pri tvorbe sa schematický návrh importuje z nástroja Schematic Capture do editora, kde sa zobrazia odkryté časti vodivých ciest alebo piny, na ktoré sa budú upevňovať komponenty a ich označenia, ktoré budú zobrazené na doske. Pre komponenty, ktoré sa nenachádzajú v databáze, je potrebné vytvoriť vlastné kontaktné plochy podľa rozmerov v technickom hárku. V obvode sa nachádzajú dva komponenty, ktoré bolo potrebné vytvoriť: FFC konektor a snímač zrýchlenia. Piny a kontaktné plochy sú navzájom pospájané virtuálnymi čiarami, aby sa nezabudlo na prepojenie všetkých komponentov pri tvorbe ciest. Cesty môžu byť vytvorené ručne, alebo pomocou funkcie AutoRoute, ktorá ich podľa predvolených, alebo nami nastavených parametrov automaticky vygeneruje.

Ako prvé som začal s návrhom základnej dosky, pre ktorú bolo potrebné vytvoriť ohraňenie. Základná doska má zhodný tvar a rozmiestnenie pinov s doskou Arduino UNO, ktorej rozmery sú verene dostupné. Do oblasti dosky sa následne umiestnia komponenty. Tie sa môžu umiestniť buď na vrchnú, alebo spodnú časť dosky. Na vrchnej strane dosky

sa musí nachádzať potenciometer a konektor pre pripojenie snímača. Zvyšné komponenty som sa snažil umiestniť tiež na vrchnú časť, predovšetkým z toho dôvodu, aby nedošlo ku kontaktu s komponentami na doske Arduino. Medzi problematické miesta patrí najmä USB konektor, napájací konektor a samotný mikrokontrolér.

Pri umiestnení potenciometra bolo potrebné vybrať miesto, do ktorého nezasahuje rameno pri otáčaní a keďže ide o komponent s nožičkami, ktoré prechádzajú cez dosku, tak aj miesto, kde z opačnej strany sa nebudú dotýkať spomínaných častí.

Servomotor je na doske umiestnený vo výreze, ktorý odpovedá jeho rozmerom s mierou rezervou potrebnou pre montáž. Pre jednoduchší návrh umiestnenia bol výrez vytvorený ako samostatný komponent, ktorého súčasťou sú aj diery pre skrutky, s ktorými sa servomotor pevne prichytí na dosku. Poloha dier pre skrutky je presne vymeraná podľa rozmerov komponentu. Umiestnenie výrezu bolo zvolené podľa voľného miesta na doske Arduino tak, aby nedošlo ku kontaktu so zvyšou elektronikou. Servomotor je napájaný a ovládaný prostredníctvom troch vodičov, ktoré sú v konečnej verzii skrátené a priamo prispájkované do dier, na ktoré môže byť pri prvých testoch umiestnený konektor, do ktorého sa servomotor jednoduchšie napojí. Miesto pre pripojenie som vyberal čo najblížie k servomotoru z tej strany, z ktorej z neho vystupujú vodiče. Komponenty, ktoré súvisia s napájaním servomotora, kondenzátor a dióda, sú umiestnené medzi konektorom na servomotor a napájacími pinmi tak, aby vodivé cesty mali čo najmenšiu dĺžku.

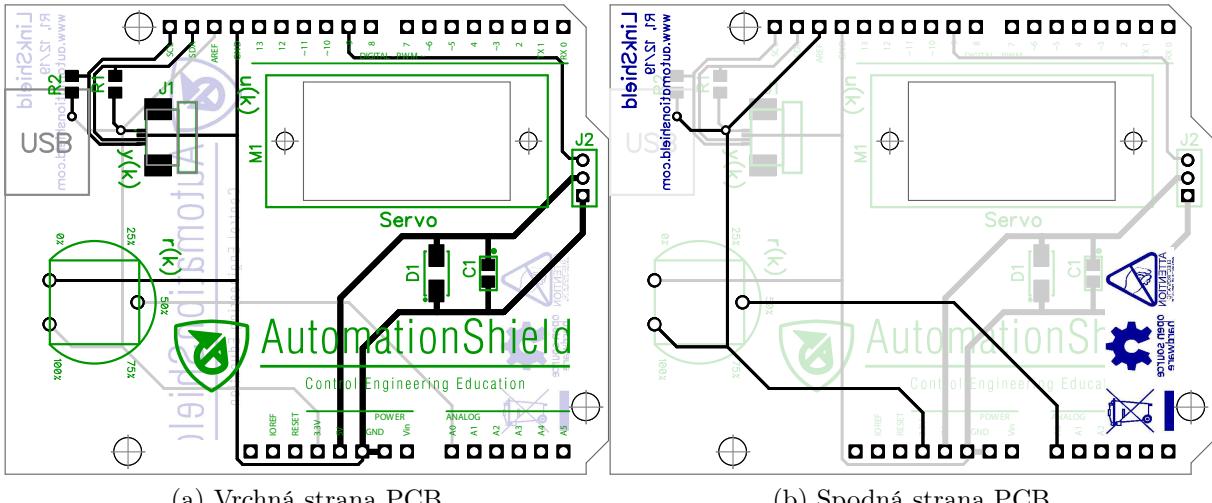
Spolu s umiestnením servomotora súvisí aj umiestnenie konektoru pre snímač. Ten je umiestnený tak, aby pri strednom uhle natočenia ramena na servomotore bol presne z opačnej strany, teda vo vertikálnej rovine s osou ramena. Konektor je orientovaný vstupným otvorom smerom k servomotoru.

Posledné dva komponenty, ktoré bolo potrebné na dosku umiestniť boli pullup rezistory. Tie sú umiestnené medzi konektorom na snímač a I2C pinmi na doske.

Pre tvorenie ciest som zvolil ručný spôsob, keďže obvod nie je náročný. Šírka väčšiny ciest je podľa predvolených parametrov, čo je 0,33 mm. Pre cesty, ktoré napájajú servomotor som zväčšil hrúbku na dvojnásobok pre možný väčší prúd prechádzajúci touto časťou obvodu. Hrúbka dosky je rovnaká ako pri doske Arduino, teda 1,6 mm.

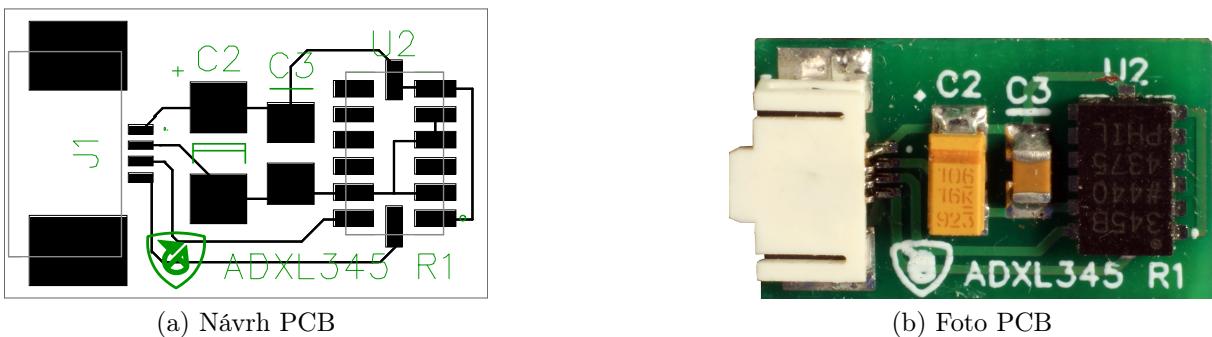
Pri vytváraní ciest sa podľa mnohých publikácií a literatúry uvádza, že v miestach s pravouhlým zakrivením ciest dochádza k elektromagnetickej indukcii - EMI a odporúčalo sa použiť väčšie uhly. Taktiež aj výrobcovia v minulosti odporúčali vyhnúť sa pravouhlým rohom z dôvodu staršej technológie výroby, pri ktorej neboli cesty dostatočne presné a na rodoch mohli vzniknúť chyby, ktoré by spôsobovali rušenie signálu. V súčasnosti je už technológia výroby omnoho presnejšia a je publikovaných mnoho výskumov ktoré vyvracajú tvorenie EMI pri bežne používanej komunikácii. Podľa publikácie Marka I. Montrosea na medzinárodnom sympóziu o elektrickej kompatibilite [11] tvrdí, že ani pri frekvenciach nad 1 GHz nebolo možné zaznamenať rozdiel medzi 135° a 90° zakrivením. V mojom návrhu som používal 135° zakrivenia, ale je to primárne z dizajnových dôvodov ako z funkčných. Najvyššia frekvencia pri komunikácii medzi senzorom a Arduinom prostredníctvom I2C komunikácie je nastavená na 100 kHz alebo 400 kHz.

Tvorenie plošného obvodu pre snímač bolo o niečo jednoduchšie pre menšie množstvo komponentov. Na tejto doske sa nachádzajú štyri komponenty, konektor, dva kondenzátory a snímač. Cieľom pri tvorení tohto obvodu bolo dosiahnuť najmenšiu možnú veľkosť. Komponenty sú usporiadané rovnobežne za sebou a medzery medzi nimi sú nastavené na najmenšie hodnoty podľa výrobných schopností výrobcu. Šírka ciest bola ponechaná na



Obr. 1.7: Návrh plošného obvodu základnej dosky

0,33 mm. Po minimalizovaní všetkých odstupov som vytvoril ohraničenie dosky. Finálne rozmery obvodu sú  $14,84 \times 8,89$  mm. Hrúbka dosky bola zvolená 0,2 mm ako najmenšia z ponuky výrobcu, pre ktorú platí ešte štandardná cena.

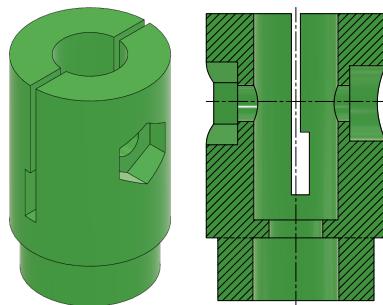


Obr. 1.8: Rozhranie akcelerometra

### 1.3 Upevnenie ramena

Na upevnenie ramena k servomotoru je potrebný medzikus, ktorý efektívne prenesie krútiaci moment. Pre tento účel je výhodou konštrukcia hriadele servomotora, na ktorej sú vytvorené drážky a uprostred je vywŕtaný závit. Vzhľadom na časovú efektívnosť a ľahkú dostupnosť 3D tlače, bolo najefektívnejšie vymodelovať a vytlačiť si vlastný model držiaka, ktorý je zobrazený na Obr. 1.9.

Pri tvorení 3D modelu som sa snažil dosiahnuť čo najmenší tvar, ale zároveň dostatočne pevný. Keďže ide o komponent, ktorý sa otáča, zvolil som valcovitý tvar. Uprostred valca je vytvorená diera pre skrutku, s ktorou sa držiak prichytí k hriadieli servomotora. Zo spodnej časti je vytvorený výrez, do ktorého zapadne drážkovaný hriadeľ. Pri prvom návrhu som vytvoril drážky aj do spodného výrezu, lenže presnosť používanej 3D tlačiarne



Obr. 1.9: Držiak ramena.

nebola dostatočná na to, aby sa premietli aj do reálneho modelu. Pri ďalších modeloch som vytvoril len kruhový výrez s priemerom o niečo menším ako priemer hriadeľa. Kedže materiál používaný na 3D tlač je pomerne mäkký pri osadený sa drážky hriadele zaryli do stien výrezu a držiak držal dostatočne pevne. Na uchytenie ramena je stredom vedený zárez, do ktorého sa rameno vsunie a z bočných strán sú vytvorené diery pre skrutku a maticu, ktorými sa stiahne medzera a rameno sa pevne prichytí. V spodnej časti výrezu je vidieť menšie rozšírenie, ktoré je určené na uloženie vodiča. Na tlač komponentu bola použitá 3D tlačiareň Prusa i3 MK3/S.

## 1.4 Vyhodenie

Záverečná časť tvorby hardvéru bolo poskladanie do funkčného stavu. Najťažšou časťou celého procesu bolo ručné spájkovanie. Pri spájkovaní bolo nutné použiť rôzne metódy v závislosti od typu komponentu. Ako prvé som začal s najzložitejšími komponentami, medzi ktoré patria FFC konektory a snímač.

### 1.4.1 Pájkovanie

Akcelerometer, ktorý som sa rozhodol použiť má kontaktné plochy zo spodnej strany. Z hľadiska finančnej úspory bolo výhodnejšie zakúpiť snímač umiestnený v prototipizačnom obvode ako bolo znázornené na Obr. 1.5b, ako len samostatne. Preto bolo pred spájkovaním potrebné snímač najprv oddeliť od obvodu. Na oddelovanie som použil horúci vzduch, pretože ani s mikrospájkovačkou sa nebolo možné dostať ku kontaktným plochám. Z odspájkovaného snímača bolo potrebné odstrániť prebytočný cín, čo som docielil pomocou medeného odsávacieho lanka, do ktorého sa po nahriatí spájkovačkou absorbuje väčšina spájky. Na naspájkovanie na nový obvod som použil rovnakú metódu ako na odspájkovanie. Pred upevnením snímača na dosku bolo potrebné naniestť nový cín na kontaktné plochy. Na tento účel som použil spájkovaci pastu, ktorá je tvorená z dvoch hlavných príasad, gélového tavidla a spájkovacej zliatiny vo forme mikroguličiek. Zmes sa rovnomerne naniesie na kontaktné plochy a pôsobením horúceho vzduchu sa zliatina spojí do jedného celku a vďaka tavidlu sa rovnomerne rozloží po kontaktnej ploche. Následne stačí priložiť snímač na presné miesto a rovnomerne nahrievať oblasť, aby sa prepojili všetky kontakty. Pri tejto metóde spájkovania bolo potrebné byť obzvlášť opatrnlý, pretože sa nahrieva celá súčiastka a pri príliš vysokých teplotách by sa mohla trvalo poškodiť. V technických špecifikáciách sú pri väčšine komponentoch uvedené teploty ktoré sú schopné zniest', v

detailnejších sú zobrazené aj profily, podľa ktorých je odporúčané spájkovať. Z profilu možno vyčítať rýchlosť ohrevania, maximálnu teplotu a maximálny čas, pri ktorom je možné udržať teplotu komponentu na danej teplote bez poškodenia.

Pri upevňovaní FFC konektoru som zvolil klasickú metódu spájkovania. Konektor okrem štyroch kontaktných pinov disponuje ďalšími dvomi upevňovacími pinmi, ktoré sa nachádzajú na bokoch. Pre tieto piny sú na doske umiestnené kontaktné plochy ku ktorým je možné ich prispájkovať. Ako prvý som prichytil konektor pomocou bočných pinov, aby pri upevňovaní kontaktných pinov nedošlo k posunutiu konektora a nesprávnemu prepojeniu kontaktov. Medzery medzi kontaktnými pinmi sú veľmi malé a pri väčšom množstve spájkovacieho kovu môže dôjsť k ich vzájomnému prepojeniu. V prípade ich prepojenia je možné kov odstrániť pomocou medeného odsávacieho lanka ako v prípade snímača. V tomto prípade na kontaktoch zostane dostatočné množstvo kovu a nie je nutné opäťovné nanášanie.

Po konektore a snímači nasledovali SMD komponenty. Na tie som zvolil tiež klasický spôsob spájkovania. Všetky zvyšné SMD komponenty majú len dva konektory a presnosť nie je až tak podstatná ako pri predchádzajúcich dvoch. Pri umiestňovaní tantalových kondenzátorov a diódy je potrebne dávať pozor na správnu orientáciu pretože sú polarizované. Na uľahčenie sú na doske vytlačené značky, ktoré naznačujú stranu na ktorej sa má nachádzať katóda na kondenzátore a pri dióde je označený prieplustný smer. Rezistory a keramické kondenzátory nie sú polarizované a nezáleží na orientácii vývodov.

Posledné komponenty, ktoré je potrebné prispájkovať k doske sú konektory a potenciometer, ktoré majú nožičky. Pre tieto komponenty a sú na doske umiestnené diery, cez ktoré sa kontakty prestrčia a prispájkujú sa zo spodnej strany. Použité konektory slúžia na upevnenie k doske Arduino. Z vrchnej strany majú dutinky ku ktorým sa môžu pripojiť ďalšie elektronické komponenty. Zo spodnej strany sú predĺžené kolíky, ktoré zapadnú do dutinkových konektorov na Arduine.

#### 1.4.2 Servomotor, rameno a snímač

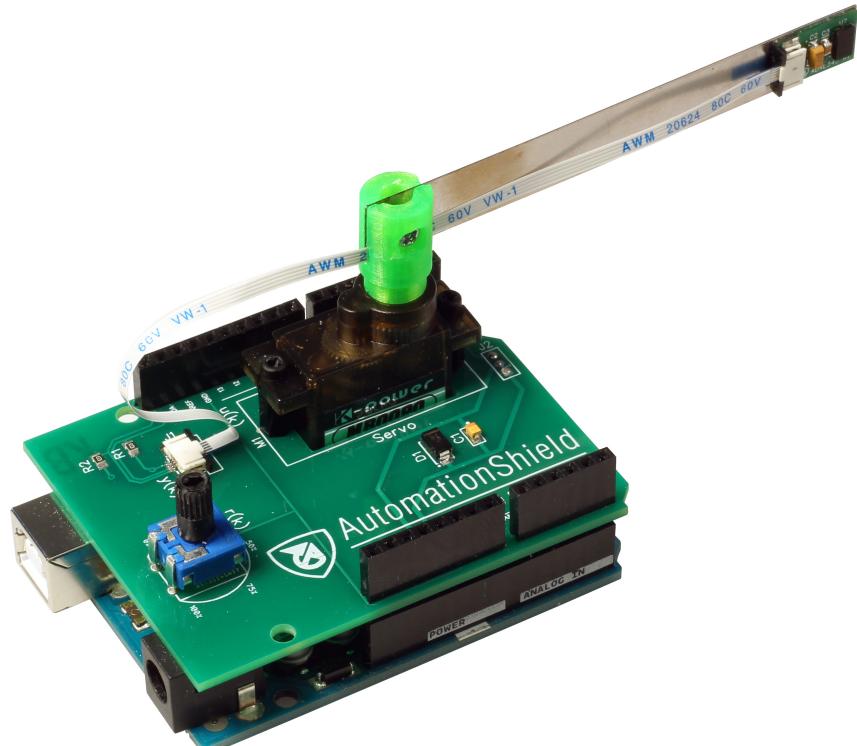
Servomotor je k doske pripojený skrutkami, maticami a vymedzovacími podložkami cez predvŕtané diery. Výška servomotora je podstatne väčšia ako hĺbka pripraveného otvoru po nasadení LinkShieldu na Arduino. Na zdvihnutie servomotoru do vhodnej výšky som použil 10 mm vymedzovacie podložky. Pre prípadné testovanie iných modelov môžu byť použité vymedzovacie podložky s rôznou výškou.

Obvod snímača je umiestnený na konci ramena. Prichytenie obvodu k ramenu je prostredníctvom adhézneho lepidla B-7000 na báze silikónu. Toto lepidlo dosahuje extrémne vysokú adhéziu k plastovým povrchom a zachováva si pružné vlastnosti. Zvyčajné použitie tohto lepidla je na spájanie a opravovanie dielov prenosnej elektroniky ako napríklad dotykové displeje. Dôvod výberu adhézneho lepidla je ten, že lepené povrhy, či už rameno alebo samotný plošný obvod, sú ľahko leptateľné. Pri kyanoakrylátových lepidlách dokážeme dosiahnuť podobne vysokú adhéziu ale sú typické tvorením bieleho povlaku v okolí spoja a spoj je krehký a môže sa poškodiť pri vibrovaní ramena.

Rameno sa upevňuje do už spomínaného držiaku. Držiak sa najprv upevní k hriadeľu servomotora pomocou skrutky, ktorá bola súčasťou balenia servomotoru. Rameno sa následne vsunie do štrbiny a upevní sa skrutkou a maticou cez otvory v držiaku a ramene.

Skompletizovaný LinkShield je zobrazený na Obr. 1.10. Na Obr. 1.10a je fotografia

celej zostavy, na Obr. 1.10b je detailnejší pohľad na základňu modulu a na Obr. 1.10c je detail obvodu snímača umiestnenom na ramene.



(a) Izometrický pohľad



(b) Pohľad z vrchu



(c) Detail ramena

Obr. 1.10: Vyhotovenie modulu.

## 1.5 Cena

Jedna z hlavných priorít projektu AutomationShield, ktorého súčasťou je aj LinkShield, je cenová dostupnosť. Modul je navrhovaný tak, aby pre každého kto sa ho rozhodne zstrojiť, bol cenovo dostupný. Najdrahším komponentom, ktorý sa nachádza na module je servomotor. Zvyšné komponenty vrátene dosky majú relatívne nízku cenu. Cena celého

modulu aj jednotlivých komponentov je zobrazená v Tab. 1.1. Niektoré komponenty ako napríklad rezistory, kondenzátory, alebo diódy je možné zakúpiť len vo viacerých kusoch naraz a v ich násobkoch. Pri mnohých komponentoch zvykne byť jednotková cena pri väčších množstvách nižšia. Všetky ceny vypísané v tabuľke sú podľa jednotkových cien pri vyhotovení minimálne 10 kusov.

Tabuľka 1.1: Zoznam komponentov a cien v Eurách.

Názov	Označenie a hodnota	Popis	Ks.	Cena	Spolu
Servomotor	digitálne, kovové prevody (napr. Savox SH-0257MG)	M1	1	16.76	16.76
Akcelerometer	Analog Devices ADXL345	U2	1	0.89	0.89
Konektor	0.5 mm odstup, 4-lead FFC/FPC (napr. 52745-0497)	J1, J2	2	0.30	0.59
Potenciometer	10 kΩ, 150 mW (napr. ACP CA9MVV 10K)	POT1	1	0.10	0.10
Rezistor	0805, 10 kΩ (napr. ROYAL OHM 0805S8J0103T5E)	R1, R2	2	0.01	0.02
Kondenzátor	0805, tantalový, 4.7 µF (napr. AVX TAJP475K016RNJV)	C1	1	0.15	0.15
Kondenzátor	1206, tantalový, 10 µF (napr. T491A106M016AT)	C2	1	0.22	0.22
Kondenzátor	0805, keramický, 100 nF (napr. C0805C104M5RACTU)	C3	1	0.01	0.01
Dióda	DO214AC (e.g. Vishay BYG20J, 1.5 A, 600 V)	D1	1	0.17	0.17
Kábel	0.5 mm odstup, 4-lead FFC	-	1	0.12	0.12
PCB (shield)	2 vrstvy, FR4, 1.6 mm hrúbka, zelená maska	-	1	0.45	0.45
PCB (ADXL345)	1 vrstva, FR4, 0.6 mm hrúbka (alebo menej), zelená maska	-	1	0.45	0.45
Skrutka	M2×8, oceľ	-	1	0.02	0.02
Matica	M2, oceľ	-	1	0.01	0.01
Vymedzovač	hexagonal; polyamid; M2; 10 mm	-	2	0.15	0.30
Skrutka	M2×5, Phillips, polyamid	-	2	0.13	0.25
Matica	M2, polyamid	-	2	0.06	0.12
Oska	Oska potenciometra, (e.g. ACP CA9MA9005)	-	1	0.10	0.10
Konektor	6×1, 2.54 mm odstup	-	1	0.06	0.06
Konektor	8×1, 2.54 mm odstup	-	2	0.09	0.18
Konektor	10×1, 2.54 mm odstup	-	1	0.09	0.09
Držiak	1.1 g Zelený PETG filament, 21 m na tlač, 0.07 kWh elektr. energie	-	1	0.04	0.04
Magnety	φ9×2 mm, N50, ~ 13 N (napr. Omo Magnets N50D00960020)	-	3	0.12	0.36
Rameno	85×10×0.3 mm, φ2 mm diera 5 mm od konca, AISI 301 (S30100)	-	1	0.40	0.40
					<b>Spolu: 21.85€</b>

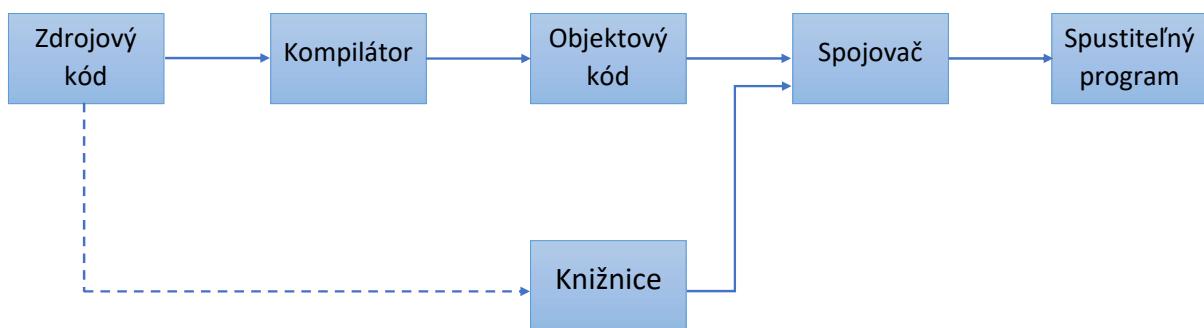
## 2 Programové rozhranie

Súčasťou tvorenia konceptu je aj vytvorenie programového rozhrania. Pri tvorení programu je dôležité riadiť sa zásadami zaužívanými pri predchádzajúcich moduloch z projektu AutomationShield. Hlavným dôvodom dodržiavania zásad je podobnosť všetkých programov a veľkou výhodou je jednoduchá orientácia medzi nimi. Pri používaní viacerých modulov postačuje zorientovať sa v jednom z programov a pri zvyšných ich štruktúra už naznačuje funkčnosť a navádza na správny postup. Súčasťou projektu je komplexná knižnica s názvom AutomationShield, ktorá obsahuje v sebe funkcie, ktoré je možné použiť pre každý z jednotlivých modulov. Samotný program je písaný v prostredí Arduino IDE, ktorý je voľne dostupný a poskytuje používateľovi prostredie vhodné pre programovanie všetkých dosiek kompatibilných s Arduino. Softvér podporuje programovanie v jazykoch C a C++, respektíve ich v kombinácii podľa špeciálnych pravidiel pre štruktúru kódu. Napísané programy sú verejne dostupné cez platformu GitHub, ktorá je postavená na systéme Git, ktorý je určený na jednoduchú správu rôznych verzíí programov. Umožňuje pracovať naraz na rôznych častiach programu súčasne viacerým vývojárom. V systéme sa uchováva história úprav a v prípade chybného kódu je možné sa kedykoľvek vrátiť späť k funkčnej verzii.

Štruktúra programu je rozdelená do dvoch častí, na zdrojový kód a príklady. Zdrojový kód je označovaný ako knižnica, ktorá zahrňa funkcie potrebné na napísanie príkladov. Knižnica zvykne pozostávať z dvoch častí, respektíve dvoch súborov nazývaných hlavička a zdroj. Hlavička slúži na deklarovanie alebo definovanie všetkých funkcií a premenných, ktoré sa majú v knižnici nachádzať. Zdrojový súbor obsahuje vytvorené funkcie zadefinované v hlavičke.

V prípade LinkShieldu bolo nutné použiť len hlavičkový súbor z dôvodu chyby kompliacie. Rovnaký problém bol zistený už pri tvorbe programu pre predchádzajúci modul s názvom BoBShield. Problém vychádza z hlavičky `AutomationShield.h`, ktorá obsahuje univerzálne funkcie, ktoré možno použiť pri ktoromkoľvek module. Nachádza sa v nej vlastná funkcia pre vzorkovanie podľa nastavenej periódy. Funkcia využíva jeden z časovačov na MCU Arduino architektúry AVR, konkrétnie `Timer1`. LinkShield a BoBShield používajú servomotor, pre ktorý je potrebná ďalšia knižnica. Servomotor je ovládaný pomocou PWM (z angl. Pulse Width Modulation - modulácia šírky impulzu) signálu, pre ktorý je potrebné využiť tiež jeden z časovačov. V prípade tejto knižnice je to tiež `Timer1`. Tým, že sa časovač využíva viackrát, dochádza k chybe pri kompliacií. Kompilácia je proces prepisovania napísaného kódu cez množstvo úkonov tak, aby ho bolo možné odoslať do procesora na doske, vo forme ktorú dokáže správne rozoznať. Na Obr. 2.1 je zobrazený zjednodušený proces kompliacie. Pri kompliacií sa zdrojový kód v súboroch s príponami `*c`, `*.cpp` alebo `*.h` odošle do komplátora. V komplátore sa súbory prekonver-

tujú z písaného jazyka C/C++ do binárneho objektového kódu. Ten je následne odoslaný do spojovača (linker), ktorý všetky súbory spojí do jedného celku. V tejto fáze nastáva spomínaná chyba, kde spojovač nedokáže správne spojiť funkcie, v ktorých sa odkazuje na časovač. Jednou z možností by bolo prepísanie vzorkovacej funkcie tak, aby používala iný časovač, táto možnosť je však komplikovaná pre relatívne veľké množstvo rôznych modulov, ktoré túto knižnicu už využívajú. Jediný spôsob ktorý fungoval, bolo spojenie hlavičkového a zdrojového súboru do jedného komplexného hlavičkového súboru, čo sa ukázalo ako vhodné riešenie pri BoBShielde.



Obr. 2.1: Zjednodušená schéma kompliacie.

## 2.1 Hlavičkový súbor

Hlavičkový (angl. header) súbor je časť programu, v našom prípade celého programu, slúžiaca na deklarovanie a napísanie funkcií dôležitých pre riadenie modulu. Názvy hlavičkových a zdrojových súborov pri projektoch AutomationShield sa zvyknú pomenovať podľa názvu modulu. Hlavičkové súby majú príponu \*.h, takže názov knižnice pre LinkShield je LinkShield.h.

Aby mohli byť funkcie použité v iných súboroch ako napríklad v hlavnom kóde, v príklade alebo tiež aj v iných knižničach, musí byť hlavičkový súbor zahrnutý v danom kóde na začiatku pomocou príkazu `#include "LinkShield.h"` alebo `<LinkShield.h>`. Úvodzovky alebo ostré zátvorky sa volia podľa umiestnenia súboru. Ak je umiestnený v predvolenom priečinku vývojového prostredia používané aplikácie, používajú sa ostré zátvorky. Tieto knižnice sú označované ako globálne a je možné ich použiť v ktoromkoľvek programe. Ak je súbor umiestnený v priečinku kde sa nachádza samotný program, ktorý tvoríme, používajú sa úvodzovky. Označujú ako lokálne a dajú sa použiť len v programe, v ktorého priečinku sa nachádzajú. Do programov sa zahrňajú len hlavičkové súby, zdrojové súby \*.c alebo \*.cpp sa zahrňujú len do hlavičkových súborov, pre ktoré sú vytvorené.

Kedže hlavičkové súby môžu byť použité vo viacerých iných súboroch jedného programu naraz, je potrebné zabezpečiť, aby sa načítali len raz. Zabezpečenie sa tvorí pomocou podmienky, do ktorej sa zabalí celý obsah hlavičkového súboru. Podľa podmienky `#ifndef` pri kompliacii komplikátor skontroluje, či už bola hlavička zadefinovaná, ak nie, tak ju zadefinuje a vytvorí značku, podľa ktorej v prípade pokusu o opakovanie načítanie rozozná. Podmienka môže vyzeráť nasledovne:

```
#ifndef LINKSHIELD_H_
#define LINKSHIELD_H_
...
//funkcie
...
#endif
```

Zo štylistickej stránky býva premenná v podmienke písaná s podobným názvom ako je názov daného súboru s tým, že miesto bodky alebo medzery sa používa podtržník. Takáto forma označovania má zabrániť výskytu rovnakých podmienok vo viacerých súboroch, kedy by nemuselo načítať jednu z hlavičiek.

### 2.1.1 Knižnice

Prvá vec, ktorá sa v hlavičkovom súbore pre LinkShield nachádza je zahrnutie externých hlavičkových súborov iných knižníc, z ktorých sú volané niektoré funkcie. Do hlavičky LinkShieldu sú zahrnuté štyri knižnice:

```
#include <Wire.h>           //kniznica pre I2C komunikaciu
#include <Arduino.h>
#include <Servo.h>           //kniznica pre servomotor
#include "AutomationShield.h" //kniznica pre AutomationShield
```

Prvá knižnica `Wire` slúži na komunikáciu prostredníctvom rozhrania I2C. Ako už bolo spomínané v Kap. 1.1.3, pomocou tohto rozhrania komunikuje Arduino s akcelerometrom. Knižnica s názvom `Arduino` je knižnica pre všetky základné funkcie používané pre Arduino ako napríklad podmienky, výpočtové operácie a podobne. Zahŕňa sa do všetkých hlavičkových a v niektorých prípadoch aj do zdrojových súborov. Dalo by sa povedať, že je to jadro celého programu. V hlavnom súbore programu \*.ino respektíve v tzv. návrhu (sketch) je táto knižnica vkladaná automaticky na pozadí. Ďalšia knižnica, ktorá je volaná sa nazýva `Servo`. Ako z názvu vyplýva, jedná sa o knižnicu určenú pre ovládanie rôznych servomotorov. Posledná knižnica ktorej funkcie budú používané v programovom rozhraní je `AutomationShield`, ktorá už bola spomínaná v súvislosti s univerzálnymi funkciami pre ostatné moduly.

Pri prvých testoch sa v hlavičke nachádzali ďalšie dve knižnice, ktoré boli určené pre nastavovanie parametrov a čítanie hodnôt na akcelerometri, `Adafruit_ADXL345_U` a `Adafruit_Sensor`. Prvá knižnica je určená presne pre konkrétny model snímača a druhá je univerzálna pre všetky snímače, ktoré daný výrobca prototypizačných ponúka. Po otestovaní správnej činnosti hardvéru boli knižnice nahradené vlastnými funkciami.

### 2.1.2 Definície

V nasledujúcej časti kódu sa nachádzajú tzv. textové definície, ktoré majú presne zadefinovanú hodnotu a pri vykonávaní programu sa nemenia. Medzi tieto definície sa radia čísla pinov na doske, ku ktorým sú pripojené komponenty. Pre čítanie hodnôt z potenciometra je zadefinovaný analógový pin 0 a pre vodič servomotora, ktorým sa ovláda natočenie je zadefinovaný digitálny pin 9. Medzi zadefinované premenne je pridaná aj adresa snímača, podľa ktorej vieme poslať príkaz do konkrétneho zariadenia komunikujúceho prostredníctvom zbernice I2C. Pre použitý akcelerometer sú k dispozícii dve adresy, 0x2D pri

napájanom pine ALT\_ADDRESS a 0x53 pri uzemnenom pine. V tomto prípade je pin uzemnený a používaná adresa je 0x53.

```
#define LINK_RPIN 0           //definovanie pinov
#define LINK_UPIN 9            //definovanie adresy pre ADXL345
#define ADXL345 0x53
```

### 2.1.3 Triedy

Jazyk C++ je objektovo orientovaný jazyk. V programe môžeme vytvoriť veľké množstvo objektov, ku ktorým priradíme atribúty a metódy, ktoré sa majú pre daný objekt vykonávať. Atribúty majú rolu premenných, ktoré objektu priradíme a metódy sú ekvivalenty funkcií. Pre atribúty a metódy sa vytvárajú triedy na základe ktorých sa volajú. Pre rôzne triedy môžu existovať metódy aj atribúty s rovnakým názvom, ktoré však môžu vykonávať odlišné úkony a mať rôzne hodnoty. Ako príklad použijem metódu `begin()`, ktorá sa používa napríklad pri inicializovaní rôznych typov komunikácie. Táto funkcia sa používa spolu s objektom, ktorý bol vytvorený pre daný typ komunikácie a používa metódy priradené danej triede. Napríklad pre sériovú komunikáciu sa používa objekt `Serial`, teda inicializácia sériovej komunikácie bude vyžerať nasledovne: `Serial.begin()`. Pre metódy v knižnici pre LinkShield je vytvorená trieda `class LinkClass{};`. Na označenie triedy sa používa kľúčové slovo `class`. Za názvom triedy sa používajú kučeravé zátvorky, v ktorých sú vypísané názvy metód a atribútov. Tie sú rozdelené do dvoch kategórií nazývaných ako špecifikátory prístupu, `public` a `private`. Rozdelenie záleží podľa dostupnosti, akú chceme atribútom a metódam priradiť. Do kategórie `public` sa vkladajú metódy a atribúty, ktoré chceme aby sa dali použiť mimo hlavičky. V kategórii `private` sa nachádzajú tie, ktoré majú byť dostupné len zvnútra objektu.

```
class LinkClass {                                //deklaracia triedy
public:
    void begin(void);                         //inicjalizacna funkcia pre LinkShield
    float referenceRead();                    //funkcia pre citanie hodnot
                                                //potenciometra
    void actuatorWrite(float);                //funkcia pre natocenie servomotora
    float sensorRead();                      //citanie hodnot zo senzora
    void calibrate();                        //kalibracna funkcia

private:
    void ADXL_POWER_CTL();                  //nastavenie meraciaho modu
                                                //akcelerometra
    void ADXL345_BW_RATE();                 //nastavenie rychlosi odosielania dat
    void ADXL_DATA_FORMAT();                //formatovanie dat
    void ADXL345_OFSZ();                   //natavenie kalibracnej odchylky
    float ADXL345_DATAZ();                 //citanie hodnot z akcelerometra na osi
                                                //Z
    int _referenceRead;                   //hodnota na potenciometri
    float _referenceValue;                //hodnota natocenia servomotora
    float _accelZ;                       //zrychlenia na osi Z
    float _angle;                        //uhol natocenia servomotora
    float _sensorBias;                  //odchylka senzora
```

```

    float _z_out;           //prekonvertovana hodnota zrychlenia
    float sensorBias(int);   //vypocet odchylyky
};


```

Pred názvom metódy alebo atribúty sa píše dátový typ výstupného argumentu. Za názvom sa v niektorých prípadoch vyskytuje zátvorka. Používa sa pri všetkých metódach, v ktorých sa vykonáva ďalší výpočet alebo iná funkcia. Pri metódach sa do zátvorky môže písť typ premennej, ktorú má očakávať, prípadne názov konkrétnej atribúty, respektíve premennej, ktorej hodnotu má používať. Ako príklad sa môže uviesť opäť funkcia `Serial.begin(unsigned long);`, ktorá očakáva vstup v dátovom type `unsigned long`, ktorý môže byť do funkcie dosadený priamo číslom alebo premenou.

## 2.1.4 Objekty

Pre hlavičku, ktorú chceme používať, musíme vytvoriť objekt, pomocou ktorého budeme metódy z danej knižnice volať. Objekt môže byť vytvorený interne priamo v danom hlavičkovom súbore alebo externe v zdrojom súbore, prípadne v programe, do ktorého je zahrnutá hlavička. Pre rozhranie LinkShieldu je vytvorený objekt vo vnútri hlavičkového súboru. V prípade, že by bol vytvorený aj zdrojový súbor, v ktorom by sa vytváral objekt, je potrebné v hlavičkovom súbore deklarovať, že sa niekde bude nachádzať. To možno dosiahnuť externým deklarovaním prostredníctvom kľúčového slova `extern`, za ktorým sa napíše názov a trieda želaného objektu.

Druhý objekt ktorý je potrebné vytvoriť je pre funkcie servomotoru. To, že si objekt vytvárame vo vlastnom kóde, poskytuje výhodu pomenovať si ho tak, ako chceme a lepšie sa orientovať v metódach najmä v prípade, že objektov pre jednu triedu používame viac.

```

LinkClass LinkShield;      //deklarovanie objektu pre LinkClass
Servo myservo;           //deklarovanie objektu pre servomotor

```

## 2.2 Metódy

### 2.2.1 Inicializácia

Prvú metódu ktorá je vytvorená má názov `begin()`. Tak, ako pri spomínamej sériovej komunikácii, aj táto metóda je inicializačná. Sú v nej vložené úlohy, ktoré sa majú vykonať len raz pri každom spustení programu. Úlohy, ktoré sú sem vložené, môžu byť vložené do hlavného programu alebo príkladu samostatne, ale aby používateľ knižnice nemusel hľadať všetko čo je potrebné pre inicializáciu, postačí mu použiť túto metódu.

```

void LinkClass::begin() {
    myservo.attach(LINK_UPIN,1000,2000);      // set Servo pin
    analogReference(EXTERNAL);                // set reference voltage
    Wire.begin();                            // initialize serial
    communication
    LinkShield.ADXL_POWER_CTL();
    LinkShield.ADXL_DATA_FORMAT();           // initialize sensor
    LinkShield.ADXL345_BW_RATE();            // set sensor BandWidth
}

```

Metóda `myservo.attach()` slúži na inicializáciu servomotoru. Môžeme vidieť, že táto funkcia sa volá pomocou vytvoreného objektu. Do metódy vstupujú tri parametre. Prvý nevyhnutný parameter určuje ktorý pin má Arduino rezervovať a používať pre posielanie signálu na ovládanie servomotoru. Hodnota je do metódy vkladaná pomocou konštanty, ktorá je zadefinovaná na začiatku knižnice na hodnotu 9. Zvyšné dva parametre sú voliteľné. Určujú rozsah dĺžok signálu PWM v mikrosekundách, podľa ktorého servomotor rozpozná, na aký uhol sa má natočiť. Táto hodnota je predvolená na rozmedzie od 544 µs pri natočení na 0° a 2400 µs pri natočení na 180°. Ak servomotor má takéto rozsahy nastavené, hodnoty sa nemusia zadávať a do funkcie postačuje vložiť len číslo pinu. Servomotor ktorý je použitý na LinkShielde má maximálny rozsah otáčania približne 150° s rozmedzím dĺžky signálu 700 až 2300 µs. Avšak pracovný rozsah, v ktorom je možné dosiahnuť maximálnu presnosť je menší a má rozsah otáčania 90°, s dĺžkou impulzu medzi 1000 až 2000 µs. Ďalšia úloha, ktorá sa v inicializácii nachádza je funkcia `analogReference()`. Podľa hodnoty vstupujúcej do tejto funkcie sa určuje referenčné napätie na vstupných analógových pinoch. Hodnota môže byť zadaná pomocou predvolených kľúčových slov, ktoré majú preddefinovanú hodnotu ako napríklad `DEFAULT` alebo `INTERNAL1V1` a pod., alebo podľa napäťia na pine `AREF` so zadaným kľúčovým slovom vo funkcii `EXTERNAL`, ku ktorému je privedených 3,3 V. Nasledujúca metóda `Wire.begin()` má za úlohu inicializovať komunikáciu cez zbernicu I2C. Metóda je volaná prostredníctvom objektu `Wire`, ktorá je súčasťou zahrnutej knižnice pre I2C komunikáciu. Zvyšné dve metódy slúžia na inicializáciu snímača ADXL345 a nastavenie parametrov snímania. Tieto metódy budú podrobnejšie opísané v ďalšej časti práce.

### 2.2.2 Metódy snímača

Nasledujúce metódy sú vytvorené pre akcelerometer ADXL345. Pri vytváraní metód som vychádzal z technického listu [2], v ktorom sú podrobne popísané všetky funkcie a možnosti, ktoré snímač ponúka. Funkcie sa na snímači aktivujú podľa hodnoty bitov v registroch. V snímači sa nachádza 58 8-bitových registrov, opisovať budem len tie, ktoré sú zakomponované v programovom rozhraní. Aby sa medzi metódami snímača dalo jednoducho orientovať, ich názvy som dával podľa názvu registra, s ktorým metóda pracuje.

Na začiatku je potrebné nastaviť na akcelerometri mód, v ktorom bude pracovať. Pre tieto nastavenia slúži register číslo 45 (0x2D), ktorý má názov `POWER_CTL`. Pre každý mód sú vyhradené bity, ktoré sú zobrazené v Tab. 2.1.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Tabuľka 2.1: Register 0x2D-POWER\_CTL

Módy, ktoré akcelerometer ponúka, sú určené predovšetkým na šetrenie energie a výpočtového výkonu MCU. Tieto módy sú v hodne najmä pre slabšie procesory alebo pre zariadenia, ktoré sú napájané z batérie a potrebujú čo najviac ušetriť energiu. Akcelerometer vie ušetriť energiu tak, že pri nízkych zrýchleniach, ktorých úroveň sa dá tiež nastaviť v inom registri, zníži frekvenciu merania podľa nastavenej hodnoty. Na LinkShielde však môžeme využiť plný výkon akcelerometra a nastaviť plnhodnotné meranie. Toto nastavenie dosiahneme nastavením bitu D3 označeného ako `Measure` na 1. Zvyšné

bity musia ostať na 0, inak by mohlo dôjsť k chybe. Zapísanie bitu na správne miesto sa dosiahne odoslaním číselnej hodnoty, ktorá vznikne spojením týchto bitov. Pri tomto registri dostaneme binárne číslo 00001000, ktoré sa môže odoslať do zariadenia v akejkoľvek číselnej sústave. Ja som zvolil zápis v binárnom tvare, aby bol prehľadný a ľahko modifikateľný.

```
void LinkClass::ADXL_POWER_CTL() {
    Wire.beginTransmission(ADXL345);
    Wire.write(0x2D);
    Wire.write(0b1000);
    Wire.endTransmission();
    delay(10);
}
```

Pre začiatok komunikácie so senzorom použijeme metódu `beginTransmission()`, do ktorého vstupuje adresa zariadenia, s ktorým bude prebiehať komunikácia. Adresu pre snímač máme zadefinovanú v konštante na začiatku hlavičky. Následne sa môžu do zariadenia odosielat' údaje použitím metody `Wire.write()`. Pre zapísanie údajov do registra zariadenia sa musí odoslať najprv číslo registra, opäť môže byť zapísané v rôznych číselných sústavách a následne sa môžu odoslať údaje pre vybraný register. Po odoslaní všetkých údajov sa ukončí komunikácia so zariadením metódou `endTransmission()`. Funkcia `delay()`, ktorá sa nachádza na konci a ma za úlohu vytvoriť časovú rezervu pre zápis údajov do snímača skôr, ako sa spustí nová funkcia.

Tento proces zápisu údajov sa opakuje pri všetkých metódach, ktoré sú vytvorené pre používaný snímač. Pri druhej metóde sa konfiguruje register 0x31 `DATA_FORMAT`, v ktorom sú uložené údaje o formáte dát, v ktorom sa majú odosielat' a tiež sa v ňom určuje merací rozsah akcelerometra.

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

Tabuľka 2.2: Register 0x31-DATA\_FORMAT

Z jednotlivých bitov znázornených v Tab. 2.2 sú pre nastavenie cieleného formátu podstatné bity D3, D1 a D0. Pri nastavení bitu D3 na 1 docielime najvyššie rozlíšenie, ktoré snímač ponúka 0,004 g/LSB. Ak by bola hodnota bitu D3 nastavená na 0, rozlíšenie pri akomkoľvek rozsahu by bolo 10-bitov. Pre použitie LinkShieldu postačuje nastaviť rozsah na  $\pm 8g$ . Pre nastavenie rozlíšenia sa používajú bity D1 a D0, do ktorých sa zapíšu hodnoty pre zvolený rozsah podľa Tab. 2.3.

Rozsah	$\pm 2 \text{ g}$	$\pm 4 \text{ g}$	$\pm 8 \text{ g}$	$\pm 16 \text{ g}$
<b>D1</b>	0	0	1	1
<b>D0</b>	0	1	0	1

Tabuľka 2.3: Modifikacia rozsahov

Bit D2 sa používa pre zarovnanie bitov výstupnej hodnoty merania. Výstupná hodnota má rozlíšenie 10 – 13 bitov, ktorá sa odosiaľa v 16-bitovom formáte, čo znamená, že ostatné niektoré bity prázdnne, respektíve ich hodnota bude 0. Pomocou bitu D2 si môžeme zvoliť,

či posielaná hodnota bude zarovnaná k ľavému okraju pri nastavení bitu na 1, alebo pravému okraju pri ponechaní bitu na úrovni 0. Zvyšné bity z registra slúžia pre spustenie testovania, výber typu komunikácie SPI medzi 4-vodičovým a 3-vodičovým a invertovanie prerušovacieho bitu na jednom z výstupov, ktorý ale LinkShield nepoužíva. Všetky zvyšné bity môžu zostať na hodnote 0 a výsledná hodnota, ktorá je posielaná do zariadenia pri opísaných nastaveniach je v dvojková sústave 00001010.

```
Wire.write(0x31);           // adresa registra
Wire.write(0b1010);         // odosielana hodnota
```

Pre snímač môže byť nastavená rôzna šírka pásma a frekvencia vzorkovania. Tieto hodnoty sa nastavujú v Registri 0x2C-BW\_RATE. Hodnoty frekvencie pre výstupné dátá, ktoré môžu byť zvolené sú v rozmedzí od 0,1 do 3200 Hz, pri čom každá hodnota v tomto rozmedzí má polovičnú hodnotu predchádzajúcej. Pre zápis hodnoty sú v registri vyhradené 4 bity, D3 - D0. Bit D4 môže byť použitý pre spustenie nízkoprúdového módu, ktorý je možné použiť len pre konkrétnie úrovne frekvencií a navyše dochádza pri ňom k vyššiemu šumu. Pri testoch som použil najvyššiu frekvenciu, ktorá sa aj osvedčila ako vhodná. Túto frekvenciu dosiahneme nastavením všetkých štyroch bitov na úroveň 1. Zvyšné bity vrátane D4 sú ponechané na 0.

```
Wire.write(0x2C);           // adresa registra
Wire.write(0b1111);         // odosielana hodnota
```

Akcelerometer disponuje registrom, do ktorého je možné zadať odchýlku, ktorú pri každej odmeranej hodnote pripočíta, až následne ju odošle. Pre každú os je vyhradený jeden 8-bitový register pre odchýlku. Pri používaní LinkShieldu sa získavajú hodnoty len pre os Z, pre ktorú je vytvorená aj kalibračná funkcia. Register pre kalibráciu osi Z je označený 0x20-OFSZ. Vstupná hodnota do registra je prekonvertovaná na dátový typ Integerw, pretože ju očakáva vo forme celočísla.

```
Wire.write(0x20);           // adresa registra
Wire.write((int) (_sensorBias)); // odosielana hodnota
```

Posledná metóda, ktorá sa týka akcelerometra, je pre čítanie dát. Hodnoty zrýchlenia, ktoré akcelerometer zosníma, uloží do šiestich registrov, dvoch pre každú os. Údaje o zrýchleniach na osi Z, ktoré potrebujeme získať sú uložene v registri 0x36 a 0x37. Aby výsledok bol korektný, musia sa údaje správne spojiť.

```
float LinkClass::ADXL345_DATAZ() {
    Wire.beginTransmission(ADXL345);
    Wire.write(0x36);
    Wire.endTransmission(false);
    Wire.requestFrom(ADXL345, 2, true);
    _Z_out = (Wire.read() | Wire.read() << 8)/256;
    return _Z_out;
}
```

Čítanie hodnôt začína inicializáciou prvého registra `Wire.write(0x36)` v ktorom sú úložné údaje. Následne sa vykoná príkaz, ktorý ukončuje spojenie. V tomto prípade do metódy vstupuje parameter `false`, ktorý spôsobí, že sa odošle údaj o reštarte ale spojenie ostane aktívne. Funkcia `requestFrom()` si vyžiada zo zariadenia požadované údaje. Do tejto metódy vstupuje adresa zariadenia, počet bajtov, ktoré sú požadované a bit, ktorý

určí, či sa má spojenie ponechať alebo ukončiť. Po tomto kroku môže Arduino očakávať 2 bajty, ktoré prečíta a spojí pomocou (`Wire.read() | Wire.read() << 8`). Dve ostré zátvorky a číslo hovoria, že druhý bajt sa ma posunúť o 8 bitov vľavo, aby prvý bajt mohol byť dosadený pred druhý. Spojená hodnota sa uloží do atribútu `_Z_out`, ktorá sa v ďalšom kroku prekonvertuje do hodnoty `g`.

### 2.2.3 Kalibrácia

Táto časť programu slúži na výpočet odchýlky na akcelerometri pri pokojnom stave. Odchýlka môže byť spôsobená nepresnosťou vo výrobe alebo naklonením hardvéru, spôsobené nerovnosťou stola. Túto metódu odporúčam použiť len v prvom prípade, keď je veľká pravdepodobnosť výrobnej chyby. V prípade nerovnosti podložky sa odchýlka mení vzhľadom na to, na aký uhol je rameno otočené. Metóda má názov `calibrate()`. Vložiť sa môže buď do inicializačnej metódy alebo do inicializačnej časti hlavného programu.

```
void LinkClass::calibrate() {
    AutomationShield.serialPrint("Calibration...");
    LinkShield.actuatorWrite(45.0);
    delay(500);
    _sensorBias = -(LinkShield.sensorBias(1000)/4);
    ADXL345_OFSZ();
    AutomationShield.serialPrint(" sucessful.\n");
}
```

Proces kalibrácie začína tým, že sa servomotor natočí do strednej polohy, čo je pri LinkShielde  $45^\circ$ . Na natočenie slúži metóda `actuatorWrite()`. Následne MCU počká pol sekundy `delay()` aby sa servomotor stihol dostať do požadovanej polohy, inak by sa mohlo stať, že by akcelerometer čítal hodnoty zrýchlenia, ktoré vznikajú vplyvom pohybu. Nasleduje výpočet odchýlky, ktorý sa uloží do premennej `_sensorBias`. Hodnota sa vypočítava v metóde `sensorBias()`, do ktorej vstupuje hodnota, ktorá určí počet vzoriek z ktorých sa má vypočítať. Výstup z tejto hodnoty sa následne odošle do metódy pre zápis odchýlky.

```
float LinkClass::sensorBias(int testLength) {
    float _accelZsum = 0;
    for (int i = 0; i < testLength; i++) {
        _accelZsum = _accelZsum + LinkShield.sensorRead();
    }
    return _accelZsum / testLength;
}
```

V metóde pre výpočet odchýlky je funkcia pre `for` cyklus, ktorá zopakuje merania podľa vstupného parametru. Získané hodnoty sa postupne spočítavajú a po skončení merania sa ich súčet vydelí počtom meraní, čím sa získa priemerná odchýlka.

### 2.2.4 Verejné metódy

Okrem metód `begin()` a `calibrate()` medzi dostupné mimo programového rozhrania patria tri ďalšie. Prvá z nich s názvom `referenceRead()` umožňuje čítať hodnoty z potenciometra. Pre získanie hodnôt sa využíva analógový pin A0, z ktorého

je čítaná hodnota napäťia príkazom `analogRead()`, do ktorého vstupuje číslo pinu. Výstupná hodnota z funkcie je v rozmedzí 0 až 1023. Prečítaná hodnota sa následne premapuje metódou `mapFloat`, ktorá je vytvorená v knižnici `AutomationShield`. Metóda mapovania sa nachádza aj medzi základnými funkciami knižnice Arduina. Rozdiel medzi týmito metódami je v použitom dátovom type čísla. Pôvodná metóda využíva čísla v dátovom type `integer`, čo sú celé čísla. Upravená metóda používa dátový typ `float`, ktorý umožnuje zapisovať desatinné čísla.

```
float LinkClass::referenceRead() {
    _referenceRead = analogRead(LINK_RPIN);
    _referenceValue = AutomationShield.mapFloat((float)_referenceRead, 0.00,
        1023.00, 0.00, 90.00);
    return _referenceValue;
}
```

Druhá verejná metóda `sensorRead()` obsahuje vo vnútri len metódu na čítanie dát zo snímača. Dôvod vytvorenia tejto metódy je podobnosť so zvyšnými programovými rozhraniami pre projekty `AutomationShield`.

```
float LinkClass::sensorRead() {
    _accelZ = ADXL345_DATAZ();
    return _accelZ;
}
```

Posledná verejná metóda je určená za zadávanie veľkosti uhlu pre natočenie servomotora `actuatorWrite()`. Do metódy vstupuje parameter s veľkosťou uhla. Hodnota sa následne pretypuje na dátový typ `integer`, pretože vstupná hodnota do funkcie servomotora podporuje len celočísla. Taktiež je hodnotu potrebné premapovať, pretože pri inicializácii servomotora sa zadali hodnoty pre otáčanie v rozsahu 90°, ale metóda berie vstupné parametre pre rozsah 180°.

```
void LinkClass::actuatorWrite(float _angle) {
    int _modAngle = map((int)_angle, 0, 90, 0, 180);
    myservo.write(_modAngle);
}
```

# 3 Identifikácia a riadenie

## 3.1 Zber dát pre identifikáciu

Identifikácia je vzorový príklad hlavného programu, zameraný na získanie prvotných informácií o správaní sa systému. Informácie o správaní sa systému sú dôležitou súčasťou modelovania dynamických dejov, ktoré sú potom dôležité pre tvorenie algoritmov alebo funkcií pre spätnoväzobné riadenie. Pri získavaní informácií sa vykoná experiment, v otvorennej slučke. To znamená, že výstupné hodnoty sú závisle čisto len na vstupných hodnotách a systém nedokáže dorovnávať vzniknuté chyby pri vykonávaní procesu, inými slovami nesledujeme požadovanú referenčnú hodnotu, ani ju neregulujeme. Pre získanie vstupno-výstupných údajov sa vykoná minimálne jedno pootočenie pri ktorom sa zaznamenávajú dátový tok z akcelerometra. Dáta môžu byť vypisované vo forme grafu alebo hodnoty cez sériový port. Pomocou externého programu môžeme tieto údaje uložiť a použiť v priďalšej identifikácii parametrov a simulácií.

Program inicializácie začína zahrnutím rozhrania pre LinkShield a vzorkovanie, ktoré je upravené pre použitie so servomotorom. Programové rozhranie vzorkovania obsahuje metódy, ktoré zabezpečia, aby rozostupy medzi jednotlivými vstupmi a výstupmi systému boli konštantné. Je to z toho dôvodu, že meranie pomocou MCU nie je spojity proces ale diskrétny. Aby sa mohli údaje porovnať so simulovaným matematickým modelom je nevyhnutné vedieť, s akou periódou sú vzorky merané.

```
#include "LinkShield.h"                                // Include the library
#include <SamplingServo.h>                            // Include sampling
```

Po zahrnutí programových rozhraní sa deklarujú premenné, v ktorých sa určí periódica vzorkovania a počet vzoriek pre jednu konštantnú sekciu experimentu. Sekcia je myšlená ako časový úsek medzi natočením servomotora na rôzne hodnoty, pripadane medzi natočením a ukončením merania pri poslednom kroku. Hodnoty krokov resp. uhlov natočenia servomotora sú zadefinované ako číselný rad `float U[] = {0.0, 90.0};`, do ktorého sú vložené uhly, ktoré chceme dosiahnuť. Zvyšné premenné sú len pomocné pre potreby výpočtu alebo zachovania binárnej hodnoty.

V ďalšej časti programu sú použité inicializačné metódy potrebné pre funkčnosť LinkShieldu. Okrem inicializačných metód spomínaných v predchádzajúcej kapitole sa tu nachádza metóda pre inicializáciu sériovej komunikácie, ktorá je nastavená na najvyššiu rýchlosť, ktorú podporuje Arduino s architektúrou AVR. Ďalšie dve metódy slúžia na inicializáciu vzorkovania.

```
// inicializacia sériovej komunikacia
Serial.begin(2000000);
```

```

// inicializacia LinkShieldu
LinkShield.begin();
LinkShield.calibrate();
// inicializacia vzorkovania
Sampling.period(Ts *1000);
Sampling.interrupt(stepEnable);

```

Premenná v metóde Sampling.interrupt() v dátovom type bool dáva informáciu, či vzorkovanie s nastavenými parametrami môže byť vykonané. Metóda vykoná kontrolu, či nie je číselný rad s hodnotami pre natočenie servomotora prázdný a následne skontroluje vzorkovaciu periódu, či je dostatočne dlhá na to, aby sa stihol vykonať aspoň jeden hlavný cyklus. Kontrola funguje na princípe podmienok, ktoré ak sú pravdivé spustia nekonečný cyklus while(1).

```

if(endExperiment == true) {
    while(1);
}
if(nextStep == true) {
    realTimeViolation = true;
    Serial.println("Real-time samples violated.");
    while(1);
}
nextStep=true;

```

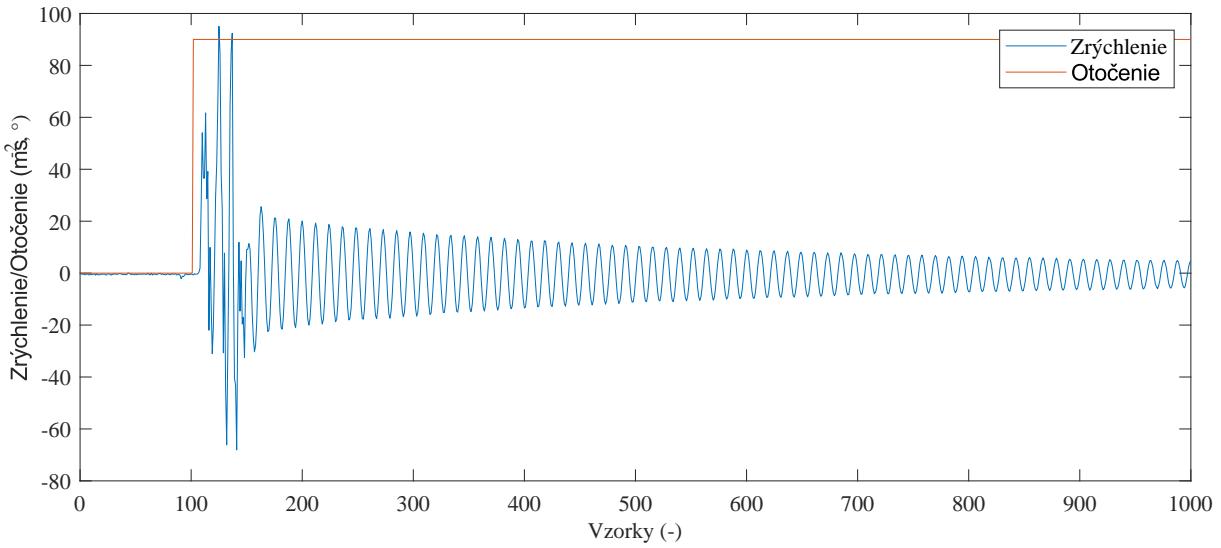
Po inicializácii sa spustí hlavný programový cyklus. V ňom sa nachádza podmienka so vstupným parametrom nextStep ktorý je v dátovom type bool, mení svoj stav vždy, keď prebehne čas pre periódu jednej vzorky. Keď je stav parametra true, podmienka je vyhodnotená ako pravdivá a vykoná vzorkovaciu funkciu. Vzorkovacia funkcia step() zabezpečuje meranie, riadenie, prepínanie medzi sekiami experimentu, teda volanie polohy servomotora z preddefinovaného číselného radu a vypnutie experimentu po vykonaní merania. Vo funkcii sa nachádzajú podmienky, z ktorých prvá porovnáva počet vykonaných sekcií s ich celkovým počtom, ktorý je rovný počtu hodnôt zadaných v číselnom rade trajektórie U. Ďalšia podmienka kontrolouje, koľko vzoriek z jednej sekcie bolo vykonaných, aby sa následne posunula na ďalší krok. Do funkcie sa následne môžu zadať príkazy na vykonanie pohybu alebo merania a taktiež aj príkazy pre odosielania údajov do sériového portu, ktorý následne môžeme monitorovať.

```

if (nextStep) {
    step();
    nextStep=false;
}

```

Po zvolení primeraných parametrov experimentu je program pripravený na spustenie. Do sériového portu sa počas vykonávania programu odosielajú hodnoty zrýchlenia ramena a uhol natočenia. Cez program určený na monitorovanie sériového portu *Real-Term* sa spustí nahrávanie, ktoré uloží odosielané údaje. Zaznamenané údaje sú vo forme tabuľky, ktorá sa použije pri tvorbe grafického výstupu a identifikácii parametrov prenosovej funkcie pri riadení. Na Obr. 3.1 je zobrazený grafický výstup, na ktorom je priebeh zrýchlení a prirodzeného tlmenia vibrácií pri otočení o  $u = 90$  (deg). Vzorkovacia períoda bola nastavená na  $T_s = 5$  ms a na konci ramena bola pridaná hmotnosť v forme troch magnetov s hmotnosťou  $m = 2,84$  g ( $\sim 0,95$  g každý).



Obr. 3.1: Grafický priebeh identifikačného experimentu.

### 3.1.1 Modelovanie

Z grafického výstupu v Obr. 3.1 je vidieť, že správanie sa ramena je z dynamického hľadiska podobné tlmenému harmonickému oscilátoru druhého rádu. Pre takéto správanie môže byť odvodený dynamický model polohy konca ramena  $q(t)$  v závislosti od času. Za predpokladu, že dynamike odozvy sústavy dominuje jedna vlastná frekvencia [5] dostaneme:

$$m\ddot{q}(t) + b\dot{q}(t) + kq(t) = F(t), \quad (3.1)$$

kde  $m$  (kg) je hmotnosť,  $b$  ( $\text{kg}\cdot\text{s}^{-1}$ ) je koeficient tlmenia,  $k$  ( $\text{N}\cdot\text{m}^{-1}$ ) je pružinová konštantá a  $F(t)$  je sila pôsobiaca na systém. Rovnicu upravíme na tvar:

$$\ddot{q}(t) + \frac{2b\omega}{2m\omega}\dot{q}(t) + \frac{k}{m}q(t) = \frac{F(t)}{m}. \quad (3.2)$$

Ak vieme, že:

$$\omega = \sqrt{\frac{k}{m}}, \quad b_c = 2m\omega, \quad \zeta = \frac{b}{b_c}, \quad (3.3)$$

a pravú stranu rovnice upravíme podľa:

$$F(t) = Cu(t), \quad C = cm\omega^2 \quad (3.4)$$

dostaneme:

$$\ddot{q}(t) + 2\zeta\omega\dot{q}(t) + \omega^2q(t) = c\omega^2u(t), \quad (3.5)$$

kde  $\omega$  ( $\text{rad}\cdot\text{s}^{-1}$ ) je vlastná respektíve prirodzená frekvencia kmitania,  $\zeta$  (-) je tlmiaci pomer,  $C$  ( $\text{kg}\cdot\text{s}^{-2}$ ) a  $c$  ( $\text{rad}^{-2}$ ) sú konštanty lineárne pôsobiaceho akčného člena a  $u(t)$  (m) je vstup do systému. Je dôležité poznamenať, že údaje, ktoré boli získané experimentom, sú údaje o zrýchlení  $y(t) = \ddot{q}(t)$ , na čo treba brať ohľad pri tvorbe riadiaceho algoritmu.

Pre návrh riadenia je dôležité správne odvodiť prenosovú funkciu, ktorá vyjadruje dynamické vlastnosti systému určené pomerom jeho vstupov a výstupov. Prenosová funkcia sa získá vykonaním Laplaceovej transformácie diferenciálnej rovnice systému. Vstup a výstup zo systému je reprezentovaný ľavou a pravou stranou. Transformáciou dostaneme obraz diferenciálnej rovnice v Lapaceovej oblasti

$$s^2Q(s) + 2\zeta\omega sQ(s) + \omega^2Q(s) = c\omega^2U(s) \quad (3.6)$$

z ktorej po úprave dostaneme prenosovú funkciu

$$P(s) = \frac{Q(s)}{U(s)} = \frac{c\omega^2}{s^2 + 2\zeta\omega s + \omega^2}. \quad (3.7)$$

### 3.1.2 Identifikácia parametrov prenosovej funkcie

Identifikácia prenosovej funkcie môže byť vykonaná v prostredí MATLAB prostredníctvom nástroja *System Identification Toolbox*. Pre najlepší odhad prenosovej funkcie je potrebné z dát získaných pri experimente vybrať oblasť, kedy je systém voľne kmitajúci. Oblast, v ktorej sa nachádza budenie môže spôsobiť nepresnosť identifikácie. Pre identifikáciu je najprv vytvorený dátový objekt `data=iddata(y, u, Ts)` v ktorom sa určia vstupné a výstupné dátá a vzorkovacia periódna. Vstupné argumenty sú údaje o polohe servomotoru  $u(t)$  a výstupné údaje sú amplitúdy zrýchlenia z akcelerometra  $y(t)$ . Vzorkovacia periódna musí byť rovnaká ako bola použitá pri meraní.

Identifikácia sa spustí príkazom `sys = tfest(dataSR, 2, 0)` v ktorom `dataSR` je objekt, v ktorom je vybratá časť z identifikácie, v ktorej sa nachádza voľné kmitanie. Číslo 2 a 0 označuje, že v prenosovej funkcií očakáva, na základe odvodenej funkcie, 2 póly a 0 núl. Po spustení identifikácie dostaneme prenosovú funkciu

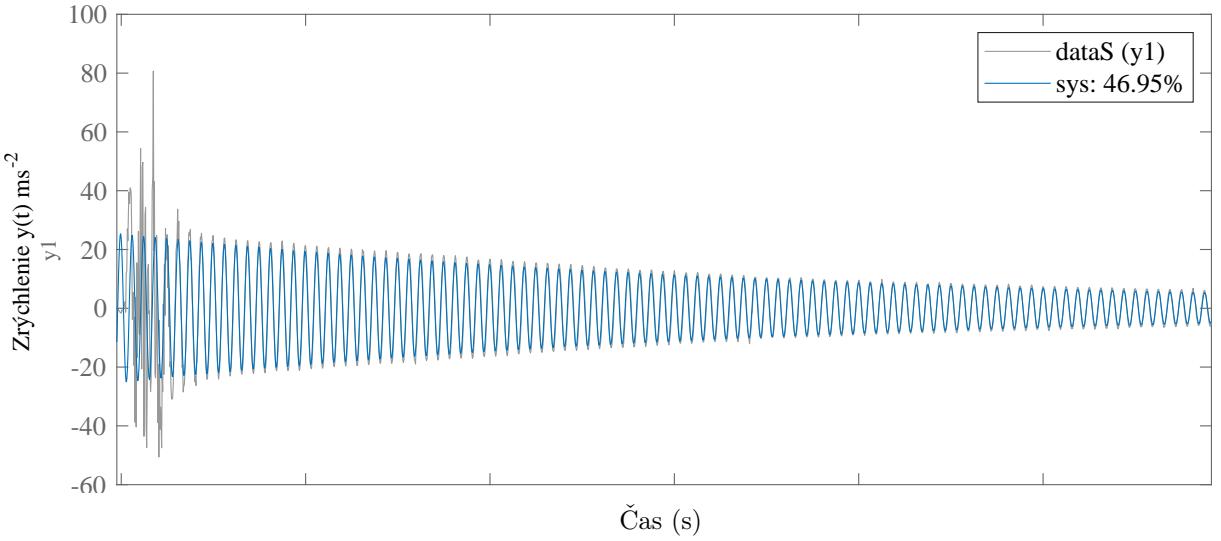
$$P(s) = \frac{9.354}{s^2 + 0.536s + 1.009e^4}. \quad (3.8)$$

Vygenerovaná prenosová funkcia sa následne môže porovnať s reálnymi hodnotami príkazom `compare(sys, dataSR)`. Pri porovnaní bola dosiahnutá zhoda  $\sim 92\%$  pre oblasť s ustáleným kmitaním. Porovnanie výsledkov je zobrazené na Obr. 3.2, na ktorom je porovnávaný rozsah aj s prechodnými javmi, čo nemôže model zachytiť z dôvodu jednoduchosti.

Z identifikovanej prenosovej funkcie je možné získať vlastnú periódnu a tlmiaci pomer, ktoré sú podstatné pre reguláciu systému. Parametre sa môžu získať použitím funkcie v MATLABe, `omega = sqrt(sys.Denominator(3))`, ktorá vyberie odmocninu z tretieho menovateľa z prenosovej funkcie, čo je v tomto prípade  $\omega^2$  a príkazu `zeta = sys.Denominator(2)/2/omega`, ktorý vyberie z druhého menovateľa číslo a osamostatní  $\zeta$ . Hodnoty ktoré boli vypočítané pre konkrétny systém sú  $\omega = 100.4 \text{ rad}\cdot\text{s}^{-1}$  (16 Hz) a  $\zeta = 0.0027$ . Tieto hodnoty je jednoduché vypočítať aj ručne na základe porovnania prenosových funkcií.

## 3.2 PPF

PPF (z angl. Positive Positioning Feedback - spätná väzba s kladnou polohou) je typ spätnovezobného riadenia, ktoré sa často používa v oblastiach s aktívnym tlmením vibrácií.



Obr. 3.2: Porovnanie reálnych a simulovaných hodnôt.

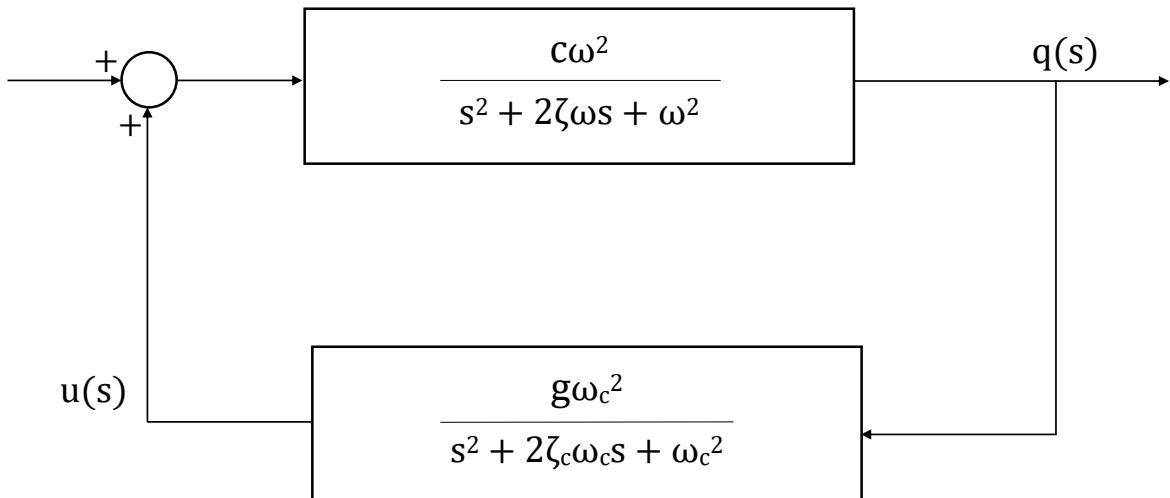
Princíp je opísaný vo vedeckej literatúre, napríklad podľa článku zameranom na aktívne riadenie prenosu zvuku, čo sú tiež vibrácie len v inej forme. V dizertačnej práci M. Yuana [15] je PPF opísané ako aktívne tlmenie pri jednej špecifickej dominantnej frekvencii na základe signálu o polohe, ktorý sa prenáša do systému druhého rádu. Systém vyhodnotí signál a zvýši tlmiaci účinok čiastočným dorovnaním okamžitej výchylky pozitívou zmenou polohy. Vstup do systému  $u_r(t)$  (deg) je tvorený referenčnou hodnotou a kompenzačnou odchýlkou  $u_r(t) = r(t) + u(t)$ . V niektorých prípadoch akým je aj LinkShield, kde je výstup meraná ako zrýchlenie ramena, je riadenie označované ako NAF (z angl. Negative Acceleration Feedback - záporná spätná väzba zrýchlenia), čo však nič nemení na základnom princípe riadenia. Pre PPF riadenie je daná diferenciálna rovnica [5, 15]:

$$\ddot{u}(t) + 2\zeta_c \omega_c \dot{u}(t) + \omega_c^2 u(t) = -g \omega_c^2 q(t), \quad (3.9)$$

v ktorej  $g$  (deg·m<sup>-1</sup>) je nastaviteľný parameter zosilnenia,  $\omega_c$  (rad·s<sup>-1</sup>) je uhlová frekvencia riadenia, ktorá sa vo väčšine prípadov nastavuje na rovnakú frekvenciu ako má systém a  $\zeta_c$  (-) je nastavovateľný tlmiaci pomer riadenia. Záporné znamienko spôsobuje prevrátenie hodnôt zrýchlenia, ktoré sú proti polohe negatívne. Z diferenciálnej rovnice kompenzátoru Laplaceovou transformáciou vznikne prenosová funkcia, ktorá sa použije pri regulácii, ako je zobrazené na Obr. 3.3:

$$G(s) = \frac{U(s)}{Q(s)} = -g \omega_c^2 \frac{1}{s^2 + 2\zeta_c \omega_c s + \omega_c^2}, \quad (3.10)$$

Do prenosovej funkcie je potrebné dosadiť optimálne parametre. Pre uhlovú frekvenciu je použitá prirodzená frekvencia ramena. Tlmiaci pomer je parameter, ktorý výrazne ovplyvňuje priebeh riadenia. Jeho hodnota sa môže pohybovať v širokom rozmedzí. Na získanie vhodného tlmiaceho pomeru pre LinkShield bola použitá simulácia spätnovázobného riadenia do ktorej boli dosadzované rôzne hodnoty tlmiaceho pomeru. Pre simuláciu vytvoríme pomocou funkcie `tf(num, den)` prenosovú funkciu algoritmu riadenia, kde `num` reprezentuje čitateľa a `den` menovateľa.



Obr. 3.3: Riadiaca schéma PPF regulátora.

```

num=-g*omega_c^2;                                % TF citatel
den=[1, 2*zeta_c*omega_c omega_c^2];            % TF menovatel
C=tf(num,den);

```

Simulácia regulovaného systému sa vykoná použitím funkcie pre spojenie prenosových funkcií späťnej väzbe `feedback(sys1, sys2)`, v ktorej `sys1` je prenosová funkcia modelu systému, ktorú sme dostali pri identifikácii a `sys2` je prenosová funkcia riadenia. Funkcia vytvorí objekt pre negatívnu spätnú väzbu, ktorý je vo výsledku tvorený jednou prenosovou funkciou. Pre simuláciu výstupu bola použitá funkcia `step(obj, __)`, do ktorej vstupujú objekty, pre ktoré chceme vykresliť skokovú zmenu a čas pre ktorý sa má vykonať simulácia.

```

cl=feedback(sys,-C);
step(90*sys,90*c1,5)
legend('Open','Closed');

```

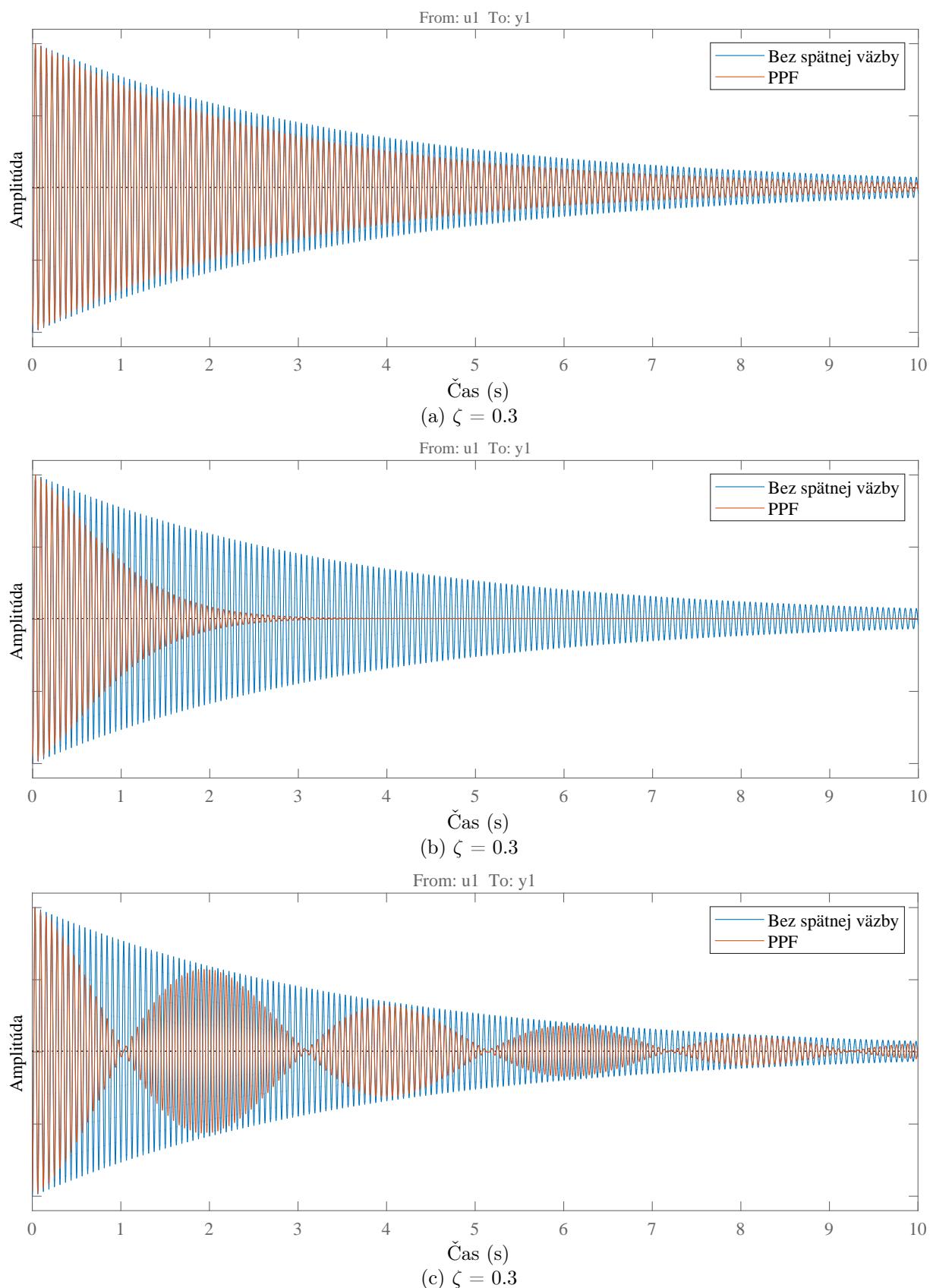
Nižšie na Obr. 3.4a až 3.4c sú zobrazené simulácie spätnoväzobného riadenia pre rôzne hodnoty tlmiaceho pomeru. Z grafického výstupu môžeme vidieť, že príliš vysoká hodnota ako je na Obr. 3.4a, kde bol použitý tlmiaci pomer  $\zeta = 0.3$  spôsobuje relatívne malé tlmenie v porovnaní s Obr. 3.4b, na ktorom je priebeh ideálny s hodnotou  $\zeta = 0.03$ . Pri menších hodnotách  $\zeta < 0.03$  dochádza po utlmení k opätovnému rozkmitaniu s menšou maximálnou amplitúdou v opačnom smere, ktoré sú zobrazené na Obr. 3.4c.

Z výsledkov simulácie sa hodnota  $\zeta = 0.03$  ukázala ako najideálnejšia a preto je používaná pri testovaní na hardvéri. Pre implementáciu riadenia na mikrokontrolér nie je možné použiť spojitú teóriu riadenia. Preto je potrebné funkciu zdiskretizovať. Pre transformáciu funkcie môže byť použitá funkcia v MATLABe `sysd = c2d(sysc, Ts)`, do ktorej vstupuje prenosová funkcia riadenia a vzorkovací čas. Po diskreditácii dostaneme charakteristickú rovnicu:

$$G(z) = -g \frac{0.1223z + 0.121}{z^2 - 1.727z + 0.9703}, \quad (3.11)$$

z čoho máme:

$$(z^2 - 1.727z + 0.9703)U(z) = -g(0.1223z + 0.121)Y(z). \quad (3.12)$$



Obr. 3.4: Simulácia s rôznymi hodnotami tlmiaceho pomeru  $\zeta$ .

Úpravou na normálny tvar diferenčnej rovnice podľa prvej differencie funkcie pre vstupno-výstupný model v diskrétnom čase  $\Delta f(k) = f(k+1) - f(k)$  dostaneme:

$$u(k+2) - 1.727u(k+1) + 0.9703u(k) = -g(0e(k+2) + 0.1223e(k+1) + 0.121e(k)). \quad (3.13)$$

Ak uvažujeme, že každá hodnota  $u(k-1)$  alebo  $e(k-1)$  je hodnota predchádzajúceho merania  $u(k)$  alebo  $e(k)$ , tak  $u(k+2)$  môžeme považovať ako výstup z regulátora, čo je zároveň aj vstup do systému.

V programe pre hardvér sa predchádzajúce hodnoty ukladajú vo forme premenných, ktoré sa po každom cykle prepíšu na predchádzajúcu hodnotu. Rovnica pre program môže byť upravená na tvar:

$$a_0u + a_1u_p + a_2u_{pp} = -g(b_0e + b_1e_p + b_2e_{pp}). \quad (3.14)$$

kde  $a_i$  až  $b_i$  sú konštantné koeficienty charakteristickej rovnice,  $u_{j,p}$  je výstup z regulátora a  $e_{j,p}$  je vstup do regulátora v závislosti od meranej vzorky. Úpravou rovnice osamostatníme výstup.

$$u = -(g(b_1e_p + b_2e_{pp}) - a_1u_p - a_2u_{pp})/a_0. \quad (3.15)$$

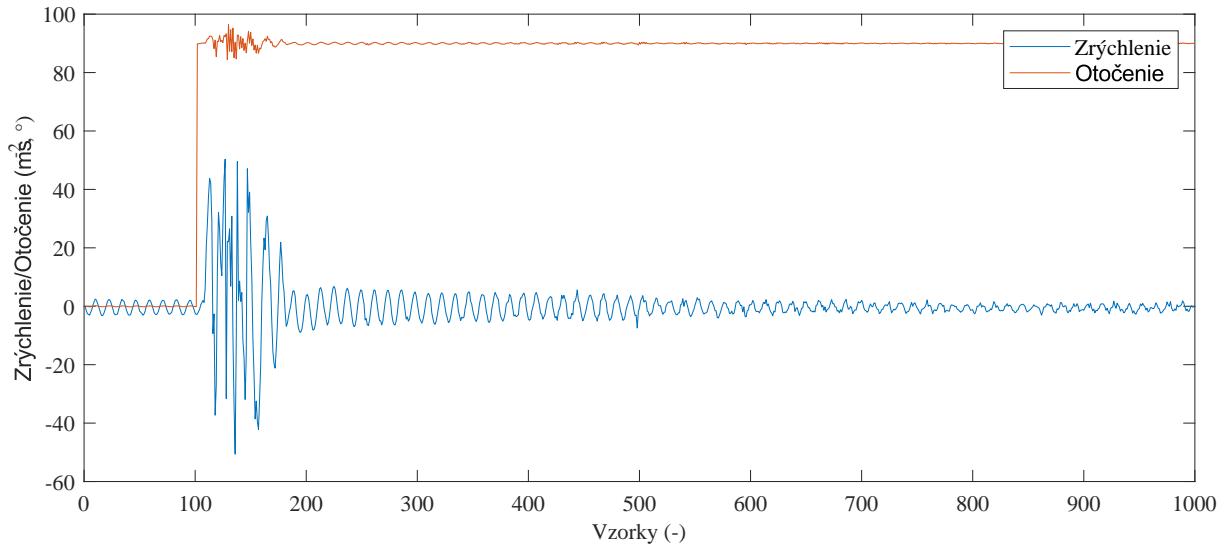
Program pre spätnovázobné riadenie pre Arduino ma rovnaký základ ako identifikačný program. Zmena je len vo funkcií `step()` do ktorej je vložená diferenčná funkcia regulátora a prepisovanie premenných na predchádzajúcu hodnotu. Zosilnenie  $g$  je nastavené na hodnotu  $g = 4$ , ktorá sa ukázala ako kritická pre čo najrýchlejšie utlmenie ramena. Vyššie hodnoty zosilnenia spôsobovali zosilnenie a nestabilitu systému.

```
u = -((g*(b1*ep+b2*pp)-a1*up-a2*upp)/a0);
e=-y;
ep=e;
epp=ep;
up=u;
upp=up;
```

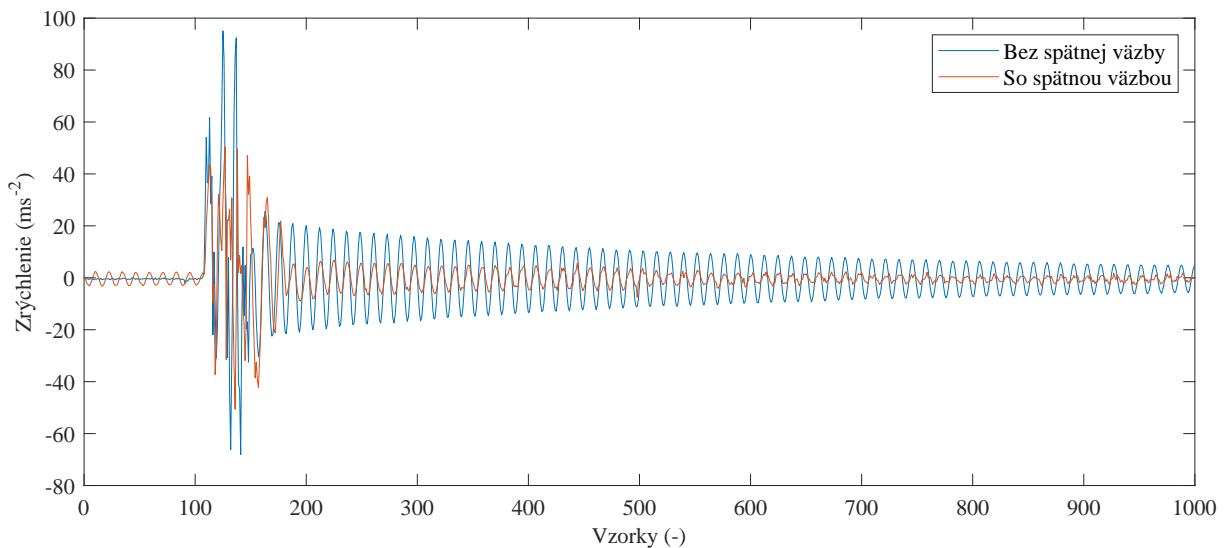
Hodnota z regulátora je pripočítavaná k referenčnej hodnote a odoslaná do servomotoru. Do sériového portu sú následne odosielané hodnoty zrýchlenia a akčný zásah.

```
LinkShield.actuatorWrite(r+u);
Serial.print(y);
Serial.print(", ");
Serial.println(u)
```

Pre vykonanie experimentu so spätnovázobným riadením sú použité zhodné parametre ako pri identifikácii, ktoré boli nastavené na otočenie o uhol  $u = 90^\circ$  pri vzorkovacej perióde  $T_s = 5$  (ms) s počtom  $s = 2000$  (-) vzoriek na jednu sekciu. Priebeh experimentu je zobrazený na Obr. 3.5a. Z priebehu je vidieť, že systém reaguje na zmeny zrýchlení a snaží sa ich kompenzovať natáčaním servomotoru. Porovnanie výstupov je zobrazené na Obr. 3.5b, na ktorom sú porovnané výstupy akcelerometra tesne po otočení ramena. Z priebehu je viditeľné rýchlejšie utlmenie voľných vibrácií, čo bol cieľ tohto experimentu.



(a) Priebeh experimentu so spätnou väzbou.



(b) Porovnanie hodnôt experimentu s voľným priebehom a so spätnou väzbou.

Obr. 3.5: Výsledky testovania PPF riadenia.

## 4 Záver

Porovnaním výstupov zo systému v otvorenej slučke a so spätnou väzbou v poslednej kapitole je možné považovať modul LinkShieldu za funkčný. Ukázalo sa, že napriek vysokým frekvenciám vibrácií je mikrokontrolér schopný na ne reagovať. Na Obr. 3.5a je viditeľné že regulátor pomáha utlmiť vibrácie ramena. Použitie takého algoritmického rozpracovania by pre prax nepostačovalo, avšak pre využíte modulu ako učebnej pomôcky je dostatočné.

Pri testovaní sa ukázali obmedzenia, ale tiež aj mnohé možnosti, ktoré môžu byť pri tvorbe novej verzie užitočné. Zásadným problémom je konštrukcia pohonného mechanizmu servomotoru, ktorá nie je dostatočne tuhá a spôsobuje problémy pri presnom otočení. Pri nízkych amplitúdach vibrácií už servomotor nie je schopný utlmoňať a musia sa nechať voľne utlmiť. Riešením by bol kvalitnejší servomotor, ktorý by bol o niečo drahší a čo by už nemuselo splňať účel modulov ako nízkonákladových alternatív ku komerčným produktom. Okrem tuhšieho a presnejšieho pohybového mechanizmu by bolo prínosné vedieť aktuálnu polohu servomotoru. Pri súčasnej verzii modulu používateľ vie len akú polohu má servomotor dosiahnuť. Všetky servomotory majú v sebe zabudované riadenie, ktoré ako vstup používa údaj o aktuálnej polohe. Niektoré z nich majú vyvedený vodič, ktorým je možné sledovať aktuálnu polohu. Menšou modifikáciou servomotorov bez tohto výstupu by sa dalo tieto údaje tiež získať.

Ďalšia zmena, ktorá môže byť použitá pri vylepšovaní je výber iného typu snímača. Výstupné hodnoty z akcelerometra súce odrážajú približný priebeh vibrácií, ale utlmiť ich správne je možné až po dosiahnutí požadovaného uhlu natočenia. Alternatívou môže byť gyroskopický senzor, ktorý dokáže merať natočenie vo všetkých osiach. Meranie natočenia je s akcelerometrom tiež možné ale len v osiach, ktoré sú vodorovné so zemským povrhom. Existujú aj snímače, ktoré obsahujú kombináciu oboch typov snímačov tzv. IMU (z angl. Inertial Measurement Unit - inerciálna meracia jednotka). Použitím takého snímačov možno implementovať iné spôsoby riadenia sústavy s tým, že by bolo možné nie len utlmiť vibrácie ale dosiahnuť aj presné zastavenie v požadovanej polohe.

Súčasná verzia však splňa hlavný cieľ práce a uvedené nedostatky treba brať ako motiváciu pre ďalší vývoj alebo skúmanie nových možností riadenia vibrácií.

# Literatúra

- [1] Adafruit. Analog accelerometer breakouts. ADXL326. [cit. 11-3-2020], <https://learn.adafruit.com/adafruit-analog-accelerometer-breakouts>.
- [2] Analog Devices. Datasheet and product info. ADXL345. [cit. 11-3-2020], <https://www.analog.com/en/products/adxl345.html#product-overview>.
- [3] Arduino AG. Arduino official store. ARDUINO UNO REV3, 2020. [cit. 10-3-2020], <https://store.arduino.cc/arduino-uno-rev3>.
- [4] General Radio Control. The difference between analog and digital RC servos. [cit. 11.3.2020], <http://www.radiocontrolinfo.com/the-difference-between-analog-and-digital-rc-servos>.
- [5] Gulán, M. and Takács, G. Automationshield online, 2020. [cit. 5-10-2020]; GitHub Wiki <https://github.com/gergelytakacs/AutomationShield/wiki>.
- [6] S. Ikizoğlu and O. Gürişik. LQR based optimal control for single-joint flexible link robot. In *2018 6th International Conference on Control Engineering Information Technology (CEIT)*, pages 1–6, 2018.
- [7] K-POWER. Mb0090. Online datasheet, K-POWER. [cit. 2020-01-24]; [https://www.tme.eu/sk/details/kp-mb0090/servomotory/k-power\(mb0090\)/](https://www.tme.eu/sk/details/kp-mb0090/servomotory/k-power(mb0090)/).
- [8] T. Kose, Y. Terzioglu, K. Azgin, and T. Akin. A single-mass self-resonating closed-loop capacitive MEMS accelerometer. In *2016 IEEE SENSORS*, pages 1–3, 2016.
- [9] D. Lim, Eun-Hye Kim, and Yong-Kwun Lee. Anti-vibration PID control for a robot manipulator experiments. In *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 724–726, Nov 2011.
- [10] K. Liu. Experimental evaluation of preshaped inputs to reduce vibration for flexible manipulator. In *2007 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2411–2415, 2007.
- [11] M. I. Montrose. Time and frequency domain analysis for right angle corners on printed circuit board traces. In *1998 IEEE EMC Symposium. International Symposium on Electromagnetic Compatibility. Symposium Record (Cat. No.98CH36253)*, volume 1, pages 551–556 vol.1, 1998.

- [12] QUANSER inc. The rotary control lab brochure. Online brožúra. [cit. 7-3-2020], <https://quanserinc.box.com/shared/static/3hhnhh25f81ionabn0yx94lm8wmgg38g.pdf>.
- [13] Savox. Sh-0257mg. Online datasheet, Savox. [cit. 2020-02-06]; <https://www.savox-servo.com/Servos-c-1338/Brushed-Motor-c-1340/Savox-Servo-SH-0257MG-Digital-DC-Motor-Metal-Gear/>,.
- [14] H. White and A. Grover. Reliability of surface-mount pptc circuit protection devices. In *Annual Reliability and Maintainability Symposium*, pages 229–236, 1997.
- [15] M. Yuan, H. Ji, W. Zhou, and R. Ohayon. Active control of sound transmission using a hybrid/blind decentralized control approach. *Journal of Vibration and Control*, 21, 12 2013.

# A Zdrojový kód programového rozhrania

```
#ifndef LINKSHIELD_H_ // ochrana pred viacnasobnym pridanim
#define LINKSHIELD_H_

#include <Wire.h> // kniznica pre I2C komunikaciu
#include <Arduino.h>
#include <Servo.h> // kniznica pre servomotor
#include "AutomationShield.h" // kniznica pre AutomationShield

#define LINK_RPIN 0 // definovanie pinov
#define LINK_UPIN 9

#define ADXL345 0x53 // definovanie adresy pre ADXL345

class LinkClass { // deklaracia triedy
public:
    void begin(void); // inicializacna funkcia pre LinkShield
    float referenceRead(); // funkcia pre citanie hodnot
    potenciometra // funkcia pre natocenie servomotora
    void actuatorWrite(float); // citanie hodnot zo senzora
    float sensorRead(); // kalibracna funkcia
    void calibrate(); // nastavenie meracieho modu

private:
    void ADXL_POWER_CTL(); // nastavenie rychlosi odosielania dat
    akcelerometra // formatovanie dat
    void ADXL345_BW_RATE(); // nastavenie kalibracnej odchylky
    void ADXL_DATA_FORMAT(); // citanie hodnot z akcelerometra na
    void ADXL345_OFSZ(); // osi Z
    float ADXL345_DATAZ(); // hodnota na potenciometri
    osi Z // hodnota natocenia servomotora
    int _referenceRead; // zrychlenia na osi Z
    float _referenceValue; // uhol natocenia servomotora
    float _accelZ; // odchylka senzora
    float _angle; // prekonvertovana hodnota zrychlenia
    float _sensorBias; // vypocet odchylky
    float _Z_out; // sensorBias(int);
    float sensorBias(int); // vypocet odchylky
};

};
```

```

LinkClass LinkShield; // deklarovanie objektu pre LinkClass
Servo myservo; // deklarovanie objektu pre servomotor

void LinkClass::begin() {
    myservo.attach(LINK_UPIN, 1000, 2000); // inicializacia servomotora
    analogReference(EXTERNAL); // nastavenie referencneho napatia
    Wire.begin(); // inicializacia I2C komunikacie
    LinkShield.ADXL_POWER_CTL();
    LinkShield.ADXL_DATA_FORMAT();
    LinkShield.ADXL345_BW_RATE();
}

void LinkClass::calibrate() {
    AutomationShield.serialPrint("Calibration...");
    LinkShield.actuatorWrite(45.0); // natocenie na stredny
    uhol
    delay(500);
    _sensorBias = -(LinkShield.sensorBias(1000)/4); // konverzia odchylky
    ADXL345_OFSZ();
    AutomationShield.serialPrint(" sucessful.\n");
}

float LinkClass::sensorBias(int testLength) {
    float _accelZsum = 0;
    for (int i = 0; i < testLength; i++) { // for cyklus pre
        zistenie odchylky
        _accelZsum = _accelZsum + LinkShield.sensorRead();
    }
    return _accelZsum / testLength;
}

float LinkClass::sensorRead() {
    _accelZ = ADXL345_DATAZ();
    return _accelZ;
}

float LinkClass::referenceRead() {
    _referenceRead = analogRead(LINK_RPIN); // citanie hodnot z
    analogoveho pinu // premapovanie hodnot na uhol
    _referenceValue = AutomationShield.mapFloat((float)_referenceRead, 0.00,
        1023.00, 0.00, 90.00);
    return _referenceValue;
}

void LinkClass::actuatorWrite(float _angle) {
    int _modAngle = map((int)_angle, 0, 90, 0, 180); // premapovanie hodnot
    pre rozsah otacania
    myservo.write(_modAngle);
}

```

```

}

void LinkClass::ADXL_POWER_CTL() {
    Wire.beginTransmission(ADXL345); // zaciatok komunikacia s ADXL345
    Wire.write(0x2D); // volba registra
    Wire.write(0b1000); // zapis hodnoty do registra
    Wire.endTransmission(); // ukoncenie komunikacie
    delay(10);
}

void LinkClass::ADXL_DATA_FORMAT() {
    Wire.beginTransmission(ADXL345);
    Wire.write(0x31);
    Wire.write(0b1101);
    Wire.endTransmission();
    delay(10);
}

void LinkClass::ADXL345_BW_RATE() {
    Wire.beginTransmission(ADXL345);
    Wire.write(0x2C);
    Wire.write(0b1111);
    Wire.endTransmission();
    delay(10);
}

void LinkClass::ADXL345_OFSZ() {
    Wire.beginTransmission(ADXL345);
    Wire.write(0x20);
    Wire.write((int) (_sensorBias));
    Wire.endTransmission();
    delay(10);
}

float LinkClass::ADXL345_DATAZ() {
    Wire.beginTransmission(ADXL345);
    Wire.write(0x36);
    Wire.endTransmission(false); // ciastocne ukoncenie komunikacie
    Wire.requestFrom(ADXL345, 2, true); // volanie dvoch bajtov
    _Z_out = ( Wire.read() | Wire.read() << 8)/256; // spojenie bajtov
    return _Z_out;
}

#endif

```

## B Identifikácia

```
#include "LinkShield.h"           // zahrnutie kniznice
#include <SamplingServo.h>        // zahrnutie vzorkovania

unsigned long Ts = 5;             // perioda vzorky v milisekundach
unsigned long k = 0;              // pomocne premenne pre vzorkovanie
bool nextStep=false;
bool realTimeViolation = false;
bool endExperiment = false;

float y = 0.0;                   // vystup
float u = 0.0;                   // vystup
float U[]={0.0, 90.0};           // trajektoria
int T = 2000;                    // dlzka sekcie
int i = 0;                       // pocitadlo sekcií

void setup() {
    Serial.begin(2000000);         // inicializacia seriovej komunikacia

    // inicializacia LinkShieldu
    LinkShield.begin();
    LinkShield.calibrate();

    // inicializacia vzorkovania
    Sampling.period(Ts *1000);
    Sampling.interrupt(stepEnable);
}

// hlavny cyklus
void loop() {
    if (nextStep) {
        step();                      // krok
        nextStep=false;
    }
}

void stepEnable() {
    if(endExperiment == true){      // ukoncenie experimentu
        while(1);
    }
    if(nextStep == true) {          // kontrola rychlosi vzorkovania

```

```

    realTimeViolation = true;
    Serial.println("Real-time samples violated.");
    while(1);
}
nextStep=true;
}

// algoritmus pre krok
void step() {

    if(i>(sizeof(U) / sizeof(U[0]))) { // vypocet a prepinanie sekcií
        endExperiment=true;
    } else if (k % (T*i) == 0) {
        u = U[i];
        i++;
    }

    y = LinkShield.sensorRead();           // citanie zrychlenia
    LinkShield.actuatorWrite(u);           // vstup

    Serial.print(y);                      // vypisanie vystupu
    Serial.print(", ");
    Serial.println(u);                    // vypisanie vstupu

    k++;                                // pocitanie vzoriek
}

```

# C PPF regulácia

```
#include "LinkShield.h"                                // zahrnutie kniznice
#include <SamplingServo.h>                            // zahrnutie vzorkovania

unsigned long Ts = 5;                                 // perioda vzorky v milisekundach
unsigned long k = 0;                                 // pomocne premenne pre vzorkovanie
bool nextStep=false;
bool realTimeViolation = false;
bool endExperiment = false;

float y = 0.0;                                       // vystup
float u = 0.0;                                       // vystup
float r = 0.0;                                       // vstup
float R[]={0.0, 90.0};                             // trajektoria
int T = 2000;                                         // dlzka sekcie
int i = 0;                                           // pocitadlo sekcií

float up = 0.0; float upp = 0.0;
float e = 0.0;  float ep = 0.0;  float epp = 0.0;

// premenne pre prenosovu funkciu PPF
float g=2;

float b0 = 0.0;
float b1 = 0.121845865729993;
float b2 = 0.120211333659273;

float a0 = 1.0;
float a1 = -1.718562876079719;
float a2 = 0.960620075468985;

void setup() {
  Serial.begin(2000000);    // inicializacia seriovej komunikacia

  // inicializacia LinkShieldu
  LinkShield.begin();
  LinkShield.calibrate();

  // inicializacia vzorkovania
  Sampling.period(Ts *1000);
  Sampling.interrupt(stepEnable);
```

```

}

// hlavny cyklus
void loop() {
    if (nextStep) {
        step();                                // krok
        nextStep=false;
    }
}

void stepEnable() {
    if(endExperiment == true) {           // ukoncenie experimentu
        while(1);
    }
    if(nextStep == true) {                // kontrola rychlosi vzorkovania
        realTimeViolation = true;
        Serial.println("Real-time samples violated.");
        while(1);
    }
    nextStep=true;
}

// algoritmus pre krok
void step() {

    if(i>(sizeof(U) / sizeof(U[0]))) {      // vypocet a prepinanie sekcií
        endExperiment=true;
    } else if (k % (T*i) == 0) {
        u = U[i];
        i++;
    }

    y = LinkShield.sensorRead();             // citanie zrychlenia

    u = -((g*(b1*ep+b2*epp)-a1*up-a2*upp)/a0); // funkcia pre PPF
    e==y;
    ep=e;
    epp=ep;
    up=u;
    upp=up;

    LinkShield.actuatorWrite(u+r); // regulovany vstup

    Serial.print(y);                      // vypisanie vystupu
    Serial.print(", ");
    Serial.println(u);                    // vypisanie hodnoty regulatora

    k++;                                 // pocitanie vzoriek
}

```

A-A

